A Lagrangian-Eulerian method for fully non-linear wave simulations using hierarchical cartesian grids with multigrid acceleration

da Silva Santos, Carlos Miguel Pereira

Award date:
2003

Awarding institution:
University of Bath

A Lagrangian-Eulerian Method for Fully Non-linear Wave Simulations Using Hierarchical Cartesian Grids with Multigrid Acceleration

by Carlos Miguel Pereira da Silva Santos

A thesis submitted in partial fulfilment
of the requirements of the degree of
Doctor of Philosophy

University of Bath

October 2003

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Para meu Pai e minha Mãe,
colunas do meu Parténon, falanges da minha mão, capítulos do meu livro

Abstract

A mixed Lagrangian-Eulerian method is proposed for the simulation of two-dimensional irrotational wave flows. The free surface is tracked through a series of massless Lagrangian particles through which the free surface boundary conditions are implemented.

The fluid domain is discretised using hierarchical Cartesian grids generated by recursive subdivision. A modified grid storage system is discussed along with the associated grid operations, such as neighbour finding and identification of empty cells. The interaction between the Lagrangian chain of surface markers and the underlying Eulerian grid is considered, leading to a novel method of computing free surface velocities.

A multigrid strategy is discussed with respect to its application to quadtree grids. An existing way of generating coarser multigrid levels is compared to a novel global approach that is found to yield better convergence acceleration. A comparative study of the performance of multigrid parameters, such as number of iterations at each level, grid visiting schedules and different prolongation operators is presented. It is shown that quadtree grids are ideally suited to a multigrid strategy when the solution of the elliptic Laplace's equation is sought, with savings of the order of 90% in CPU time with respect to a non-multigrid solution.

Grid convergence results (temporal and spatial) are presented for quasi-linear waves and compared to linear theory predictions. Results for non-linear waves are also presented and compared to second-order predictions. The code is additionally compared with other methods using a published case study and shown to be accurate.

Declaration

The Author wishes to declare that, except for commonly understood and accepted ideas, or where specific reference is made to the work of others, the content of this dissertation is his own work and has not been a result of any collaborative effort. All computer codes developed in this research are entirely his authorship. This dissertation has not been previously submitted, either in part or wholly, to any institution for any qualification.

Acknowledgements

This research was partly financed by a grant of the Engineering and Physical Sciences Research Council whose support is kindly acknowledged.

My gratitude goes in equal measure to two institutions and the people that have made them so welcoming: the Department of Mechanical Engineering of the University College London and the Department of Architecture and Civil Engineering of the University of Bath.

I am indebted for their input and guidance to the following people: Prof. G. X. Wu of University College London; to Dr. A. G. L. Borthwick and Dr. Paul Taylor of Oxford University; and to Dr. Chris Williams and Dr. Antony Darby of the University of Bath.

For their friendship and moral support that lightened the load of our parallel research endeavours, I am thankful to the community of Room 3.14 and its outpost in Room 3.09: Dr. Man Choi, Mary Claxton, Frederick Ellul, Dr. Luke Gale, Vasilios Maniatidis, Jon Shanks, Jon Shave and, the patriarch of us all, David Alfred Clark, I thank you all kindly.

However, without detriment to all the aforementioned people, the lion share of my appreciation must go to Dr. Deborah Greaves, without whose patience, advice and encouragement this work would not have been accomplished. Thank you, Deborah.

Finally, though it hardly needs mentioning, I would like to thank my family who have been indefatigable in their love and support.

List of Contents

List of Symbols

α,	angle formed by vectors **r** and **s**; sloshing wave parameter
$\boldsymbol{\alpha}$,	body acceleration vector
β	angle between surface normal and the y-axis; sloshing wave parameter
ε,	small parameter
ε_d,	discretisation error
ε_c,	convergence error
Φ, ϕ,	exact, numerical solution of velocity potential
$\hat{\phi}$,	acceleration potential
γ	initial elevation of wave; ; sloshing wave parameter
Γ_b,	fluid domain boundary defined by rigid boundaries
Γ_s,	fluid domain boundary defined by free surface
η,	wave elevation
λ,	wave length
μ,	dynamic density
ν,	kinematic viscosity
$\boldsymbol{\Omega}$,	vorticity vector
Ω_d,	fluid domain
θ,	contact angle between surface and rigid boundary
ρ,	fluid density
$\boldsymbol{\rho}$, ρ_i,	residual vector, residual term
σ,	surface tension
τ,	pseudo-time; surface tension coefficient
τ_d,	truncation error
ω_{SOR},	successive over-relaxation coefficient
ω,	wave frequency
ψ,	distance function (level-set method)
a,	fluid acceleration vector
a,	wave amplitude
A, a_i,	coefficient matrix, discretisation coefficients
b,	breadth of water tank
c,	volume fraction function (VOF method)
d,	range of interpolating kernel (SPH method)
e,	total error
f_b,	normal velocity of body b

g,	acceleration due to gravity
h,	mean water depth
i,	quadrant position
k	wave number
L,	multigrid level
l, l_b, l_m	division level, minimum division level, maximum division level
l_x,	highest division level satisfying equation (3.10)
l_y,	highest division level satisfying equation (3.11)
m	mass
M_i,	surface marker i
n,	cell reference number; normal direction
n',	binary reference number
n_r,	number of iterations before restriction
n_p,	number of iterations before prolongation
$\mathbf{n}$,	normal vector to body surface
N,	size of coefficient matrix
p,	pressure
$\mathbf{q}$, q_i,	source vector, source term
q_n, q_t,	normal and tangential surface velocities
$\mathbf{r}$,	vector connecting surface marker to cell centre
$\mathbf{s}$,	surface vector connecting two surface markers
t, Δt	time, length of time step
T	wave period
$\mathbf{u}$,	fluid velocity vector
u, v, w,	Cartesian fluid velocities
w,	cell width
W	interpolation kernel (smoothed particle hydrodynamics)
$\mathbf{x}$,	positional vector
x, y, z,	Cartesian co-ordinates
∂,	partial difference operator
∇,	gradient operator
Δ,	discrete difference; interval
D,	substantive derivative or derivative following particle motion
P,	prolongation operator
R,	restriction operator

SUBSCRIPTS

a,	atmospheric
c,	coarse grid; child cell
C1,C2,C3,C4,	corner neighbours of cell
d,	computational domain
F1,F2,F3,F4,	face neighbours of cell
f,	fine grid
i, I,	marker, cell indices
m,	wave mode
p,	parent cell
W,E,S,N,	west, east, south and north
0,	initial value

SUPERSCRIPTS

n,	iteration number
i,	quadrant position of cell
l,	division level of cell
prime,	non-dimensional value

ACRONYMS & ABBREVIATIONS

2GR,	Two Generations Removed
FD,	Finite Difference
FE,	Finite Element
FMG,	Full MultiGrid Cycle
GC,	Global Coarsening
KE,	Kinetic Energy
MAC,	Marker-And-Cell
MEL,	Mixed Eulerian Lagrangian
PE,	Potential Energy
PIC,	Particle-In-Cell
SCC,	Smallest Cell Coarsening
SOR,	Successive Over-Relaxation
SPH,	Smoothed Particle Hydrodynamics
TE,	Total Energy
VOF,	Volume-Of-Fluid

List of Figures and Tables

Chapter 2. Literature Review

Chapter3. Grid Generation

Chapter 4. Solution Method

Chapter 5. Multigrid Method

Chapter 6. Results

Chapter 7. Conclusions and Recommendations

1. Introduction

The simulation of physical phenomena involving moving boundaries has received increasing interest for the past few years due to the difficulties associated with this class of problems. These include the non-linearity of the boundary conditions at the surface and the necessity of knowing the accurate position and shape of the interface at each point in time.

Moving boundary problems are omnipresent in all branches of physics, ranging from mould filling applications to flame propagation. In this work, attention is devoted to the irrotational motion of inviscid water waves. The accurate simulation of these motions are of paramount importance in many ocean engineering contexts, ranging from the design of offshore structures, the response of floating objects or the sloshing motion of liquids in ship tanks. Whilst in some cases investigations may be carried out to a sufficient level of accuracy using linear theory, in many applications non-linear effects are of

crucial relevance. A common example of this is the interaction of the slender mooring tendons of Tension Leg Platforms (TLPs) with complex wave regimes. It is known that under certain conditions significant second- and higher-order forces occur which may cause significant oscillatory motions of the structure which can potentially endanger its structural integrity. Field data are expensive to obtain and may be unreliable. Satellite data and other imaging techniques provide only the larger-scale characteristics of sea wave behaviour, such as frequency and wavelength and are not accurate enough to allow wave loads to be accurately computed. Laboratory set-ups also incur considerable costs and, as a result, are primarily used in the final stages of development to test the behaviour of scale-models of structures and floating objects. Numerical modelling is therefore an important tool in providing insight into the non-linear behaviour of waves and their interaction with solid bodies.

When the characteristic dimension of the object to be studied is comparable to the wavelength, the fluid viscosity can be neglected as the wave-induced forces are dominated by inertia effects. In this case, the flow can be assumed inviscid and irrotational and thus accurately modelled by potential flow theory. Surface tension may also be neglected if the free surface is assumed not to undergo significant deformations. The equation that models such flows is the Laplace equation for the velocity potential. Whilst this is a linear equation, the problem is rendered non-linear by the free surface boundary conditions and a fully non-linear moving boundary method is required.

In the area of moving boundary problems, research has branched in the past 25 years into two main strategies: Eulerian techniques, such as the volume-of-fluid and level-set methods, in which the governing equations are solved on a fixed Eulerian grid and the interface is reconstructed from the data

at grid points; and Lagrangian approaches, such as Smoothed Particle Hydrodynamics, which track the position of a large number of Lagrangian particles from whose position of the surface is then determined. Whilst the former tend to suffer from a blurring of the interface due to a diffusive flux of the variables that determine the moving boundary position, the latter is computationally expensive due to the large number of particles employed, a problem exacerbated for three-dimensional problems. Other Lagrangian or mixed Eulerian-Lagrangian approaches usually employ a deformable mesh that follows the deformation of the interface as the solution progresses in time. This technique limits the choice of topology of grid cells as accurate boundary fitting is required at the interface. This in turn can require expensive grid generation procedures and consequent difficulties in implementing efficient convergence acceleration algorithms.

The other significant class of methods employed in the simulation of moving boundary problems encompasses boundary-integral type methods whereby the solution is sought at the domain boundaries alone, which entails considerable computational savings. These are, however, almost exclusively used in the simulation of inviscid flows — at which, it should be added, they have been employed with considerable success —, since inclusion of viscosity requires discretisation of the domain interior and consequent loss of the method's best features.

1.1. Aims of Research

In this work, a method that combines the best features of these types of methods is sought. Specifically, a method is developed with the aim of retaining the accuracy of a Lagrangian approach with the computational economy of a boundary integral type method whilst being able to be extended

to handle viscous flows. The ability to handle complex flow phenomena, such as wave breaking, has not been contemplated. Accordingly, a mixed Lagrangian-Eulerian method is devised, using an Eulerian grid to discretise and solve the governing equations throughout the whole fluid domain, whilst a free moving Lagrangian chain of markers is employed to track the position and shape of the moving free surface and implement boundary conditions.

The aims of this project can be itemised as follows:

- To implement a fast grid generation algorithm that provides variable refinement.
- To develop a multigrid strategy to accelerate convergence.
- To develop an accurate coupling between the Lagrangian surface mesh and the Eulerian grid.

1.2. Synopsis

The rest of this thesis is divided into six additional chapters. In Chapter 2, the literature in this area is reviewed. An overview of Lagrangian and Eulerian computational methods dealing with general moving boundary problems is presented, followed by specific techniques to tackle irrotational water waves. A review of the use of hierarchical Cartesian grids and the different techniques used to store its data is presented. Finally, a discussion of the use of the multigrid strategy, with special emphasis on its use with the grids used here, concludes Chapter 2.

Chapter 3 describes the generation of hierarchical Cartesian grids, the modified cell numbering system used here, along with the techniques devised to perform the grid operations required. The Lagrangian mesh used to model the free surface profile is also discussed, and the techniques to identify empty, full and surface cells are presented.

In Chapter 4, the equations and boundary conditions that govern the motion of irrotational water waves along with the discretisation procedure used to numerically solve them on quadtree grids are included. The iterative scheme used to solve the resultant algebraic system is discussed. Attention is devoted to the way velocities are calculated at the free surface. The inadequacy of two existing methods when applied to the current scheme is discussed, and a novel technique to calculate free surface velocities is introduced.

Chapter 5 deals with two different implementations of the multigrid strategy when applied to hierarchical Cartesian grids. These two approaches differ in the way coarser multigrid levels are generated. A novel and more global coarsening scheme is presented and compared with an existing method.

In Chapter 6, results of the simulation of standing waves are presented. Studies of grid convergence, both spatial and temporal, are included, along with a comparative study of a sloshing wave and an analysis of the conservation of wave energy. The performance of the multigrid iterations and the effect of its several parameters are analysed.

Chapter 7 concludes this thesis by gathering the conclusions highlighted by this work. Recommendations are discussed in view of the difficulties and breakthroughs experienced throughout the research. Directions into which this study may be carried are also included.

The computer code is made up of over 10000 lines of FORTRAN 77 instructions compiled and executed on an Intel-based PC using an AMD K6 processor running at 450MHz with 128MB of RAM. Post processing of data for graphical display has been performed using MATLAB 5.2.

2. Literature Review

Problems involving free surfaces are those that have their domain of interest bounded by a moving boundary whose position must be found as part of the solution procedure. Perhaps the most obvious problem that fits this description is that of the flow of water with a surface in contact with air, as studied in this research. However, the physical instances when this is the case are multiple and include capillarity, flame propagation, nuclear fusion, star formation, and countless other areas of science.

This chapter will start with an overview of the numerical methods employed in the numerical modelling of free surface flows. It will then proceed to detail specific contributions relevant to the development of the proposed solution method. It will discuss the history of quadtree grids, from their inception to their use in the field of computational fluid dynamics. Finally, previous applications of the multigrid method will be considered.

2.1. Numerical Methods for Flows with Moving Boundaries

This survey will cover the various methods that have been devised to model flows of Newtonian fluids with moving boundaries. In the first two subsections, methods have been categorized according to their mathematical formulation, regardless of the class of flows they are applied to. The third subsection deals with methods specific to irrotational flows.

For flows with a free surface, two boundary conditions are used at the interface:

- the dynamic boundary condition that requires equilibrium of forces, stating that the normal forces on either side of the free surface are of equal magnitude and opposite direction, while the forces in the tangential direction are of equal magnitude and direction;
- the kinematic boundary condition that expresses the fact that the free surface is a sharp boundary between the fluids – a material interface – by stating that the normal component of the fluid velocity is the normal component of the velocity of the free surface, i.e. there is no flow through the interface.

Due to the non-linearity of the free surface boundary conditions, an analytical solution cannot be obtained without simplifying assumptions. A common simplification is to convert the problem into a fixed boundary problem when the expected deformation of the interface is small using domain perturbation or small deformation theories. When the deformation is not small, a numerical procedure must be employed. The main difficulties are:

a) the boundary conditions at the free surface are non-linear;

b) the solution domain has an irregular and changing geometry, which may develop highly deformed regions and even break up into different regions which can then cross and remerge;

c) the shape of the interface must be known accurately to account properly for surface tension effects;

d) the presence of singularities at the intersection of the free surface with solid walls poses additional difficulties.

Numerical algorithms developed to solve free surface flows can generally be divided into three categories: Eulerian, Lagrangian and mixed Eulerian-Lagrangian.

Eulerian methods are characterized by a co-ordinate system that is either stationary or moving in a prescribed manner that is not directly coupled to the fluid motion. It follows that fluid travels between cells in the Eulerian grid. By contrast, Lagrangian methods employ a co-ordinate system that moves with the fluid. Consequently, a computational cell contains the same fluid particles from one moment in time to the next. Mixed Eulerian-Lagrangian schemes attempt to retain the advantages of each approach whilst reducing their disadvantages.

2.1.1. Lagrangian Methods

The Lagrangian approach offers two main advantages when applied to the simulation of free surface flows: a) it allows interfaces to be precisely delineated and followed; and b) it permits the free surface boundary conditions to be applied with ease (the latter advantage partially being a consequence of the former). However, long time simulations coupled with large flow

deformations tend to originate highly distorted meshes and resulting inaccuracies. As a result, strictly Lagrangian methods are limited to short simulation time intervals, so as to prevent mesh tangling. Hirt *et al.* (1970) propose such a scheme (LINC, Lagrangian Incompressible) for the simulation of two-dimensional incompressible viscous flow with a free surface. It employs a finite-volume discretisation on quadrilateral cells. The free surface is tracked by aligning a series of cell vertices with the interface at the start of the simulation. The implementation of the pressure condition at the free surface requires additional cells outside the free surface. Butler (1971) discusses extensions to the LINC method that include the treatment of surface tension and a mechanism to retard grid distortion based on a model of Hooke's Law force with damping.

When meshes become highly distorted, Lagrangian methods usually employ one of two techniques: mesh rezoning (or remapping) or mesh reconnection. It is worth mentioning that the criteria for judging if a mesh is to be rezoned or reconnected are not always obvious and have so far lacked rigorous justification. As a result, both techniques are either continuously applied at each time step or at fixed intervals of the simulation time.

Mesh rezoning consists of introducing a new mesh and transferring information from the old mesh onto it. Hirt *et al.* (1974) propose an arbitrary Lagrangian-Eulerian (ALE) method, based on the discretisation scheme of the LINC method, capable of dealing with high speed flows by virtue of continuous rezoning. The method is of mixed character since, during the rezoning procedure, the fluid velocity may be higher than that of the mesh. The interface is tracked in the same manner as in the LINC method. Consequently, the ALE code also suffers from limitations to the amount of distortion the free surface may undergo, though these are less severe than for

its strictly Lagrangian counterpart. This is due to the need of preserving node connectivity and the quadrilateral topology of grid cells.

Rezoning procedures and the associated transfers of mass, momentum and energy quantities between the affected computational cells carry with them an undesired convective fluxing associated with Eulerian methods, caused by the numerical diffusion that the multiple interpolations required generate. This can be particularly severe when continuous rezoning is employed. Pert (1983) discusses the use of high order approximations for the flux in order to reduce such an effect. Dukowicz (1984) and Dukowicz & Kodis (1985) improve the accuracy of the rezoning process by employing a conservative formulation for the transfer of conserved quantities between computational cells. Floryan & Rasmussen (1989) mention the CAVEAT code of Addessio *et al.* (1986), for the simulation of time-dependent, multiphase, compressible flows, which includes the above refinements to the rezoning of the interior of the computational domain, whilst using tangential and normal remapping of the Lagrangian boundary vertices, which simplifies the treatment of strongly distorted surfaces. Topological restrictions, such as those of quadrilateral cells in the ALE code, are overcome using Voronoi-Delaunay meshes (Voronoi (1908), Delaunay (1934)), as employed by Dukowicz (1981) to remap two grids of different topology.

A more radical approach to the problem of dealing with highly distorted Lagrangian meshes is introduced by Crowley (1971), called the Free Lagrangian method or FLAG. In it, mesh connections between nodes are not constant throughout the solution and mesh points can be deleted, added or reconnected at each time step. To minimise topological constraints the mesh points are reconnected to form triangular computational cells. Different criteria for the location of mesh point neighbours together with algorithms for

the generation of these grids are described by Fritts *et al.* (1985). In order to have well defined interfaces, the position, velocities and accelerations are node centred, whilst energy and mass are cell centred. As a result, reconnections tend to mix quantities of adjacent cells which introduces diffusion.

The Lagrangian approach has received considerable interest in fluid dynamics in a different class of methods, called particle methods, which define the state of the fluid system by the properties of a finite cluster of particles and simulate the dynamics of the flow by modelling fluid molecular forces. There have been two different ways of implementing this concept: purely Lagrangian particle-particle schemes, and Lagrangian-Eulerian particle-mesh methods.

Purely Lagrangian particle schemes are based on the idea that fluid motion can be simulated using a molecular model that mimics pressure from the inter-particle forces. Initial efforts were devoted to plasma simulation and astrophysical phenomena as reviewed by Hockney & Eastwood (1981). These included the particle-and-force (PAF) method of Daly *et al.* (1964). Only neighbouring particles are permitted to interact with any individual particle, which requires particle methods to include neighbour finding routines often coupled with an auxiliary mesh. The major uncertainty affecting these methods is whether the assumed viscosity models are physically realistic. Furthermore, incompressibility is of difficult implementation and the determination of spatial derivatives is often plagued by the inaccuracies associated with awkward distributions of the neighbouring particles.

Recently, a purely Lagrangian particle method — called smoothed particle hydrodynamics (SPH) —, initially proposed by Lucy (1977) and Gingold & Monaghan (1977) in the context of astrophysical problems, has received increasing attention in fluid dynamics. Monaghan (1992) discusses

the application of SPH to various fields of physics including CFD, and specifically to free surface problems (Monaghan (1994)). Its major advantage in relation to other particle schemes lies in the use of an interpolation kernel that is incorporated in the governing equations. The spatial derivatives can then be obtained from the analytical differentiation of the interpolation formulae, smoothing the numerical noise present in other particle methods. The divergence of the velocity field, $\mathbf{u}$, for example, is expressed in terms of the interpolating kernel, W, at particle a,

$$(\nabla \cdot \mathbf{u})_a = \sum_b m_b \mathbf{u}_b \cdot \nabla W_{ab}, \tag{2.1}$$

where m_b is the mass of particle b, W is the interpolation kernel and the summation is over the neighbouring particles of particle a. Several interpolation kernels are proposed by Monaghan (1992) and Belytschko *et al.* (1998). The most widely used is the Gaussian kernel (Monaghan (1994), Warren & Salmon (1994)) which, in two-dimensions, has the form,

$$W_{ab} = \frac{\exp\left(-|\mathbf{x}_a - \mathbf{x}_b|^2 / d^2\right)}{\pi d^2}, \tag{2.2}$$

where d is a parameter controlling the range of the kernel, i.e. the relative influence of neighbouring particles to the interpolation at a, and $\mathbf{x}$ is the positional vector of each particle. The kernel function (2.2) tends to a Dirac distribution as $d \to 0$, as illustrated in Figure 2.1. SPH is usually a compressible Lagrangian method since formal implementation of the constraint of constant density results in cumbersome equations that cannot be solved efficiently without additional simplifications. Instead, to simulate incompressibility in a SPH formulation, the real fluid is approximated by an artificial fluid that is more compressible than the real fluid. Provided the

Mach number remains below 0.1 (Monaghan (1994)) during the simulation, the flow can be regarded as incompressible.

The main attraction of the SPH method is its robustness and its versatility in dealing with complex flow phenomena such as surface fragmentation and merging. Its disadvantages concern the difficulty in applying no-slip body boundary conditions and treating viscous effects, and the high number of particles required to obtain a meaningful solution so that one can have confidence that particles are present on the free surface. Andrillon *et al.* (2002) use SPH in a single-phase simulation of a dam breaking against a rigid obstacle, and obtain good results when compared with experimental results and numerical simulations using an Eulerian method (VOF) that models both fluids. Colagrossi & Landrini (2002) employ a two-phase SPH code to study air entrapment in the dam-break problem, and the rise of an air bubble through water. Landrini *et al.* (2001) investigate basic features of wave breaking around ships with SPH.

The other important category of Lagrangian methods comprises particle-mesh methods. In these, the solution procedure is performed in what are essentially two meshes. The field variables are computed in an Eulerian grid, whilst a set of discrete Lagrangian particles is used to simulate the material transport between the Eulerian cells. The Particle-In-Cell (PIC) of Harlow (1964) is an example of such a technique applied to compressible flow. At each time step, the internal energy of each cell is found from a conservative accounting of the energy and momentum of the particles. The internal energy is coupled with the computed density and an equation of state to yield the pressure field. The velocities at each cell are then found and used to advance the particles in the grid. It follows that the PIC algorithm uses a mixed Lagrangian-Eulerian formulation. Although the method is able to treat large

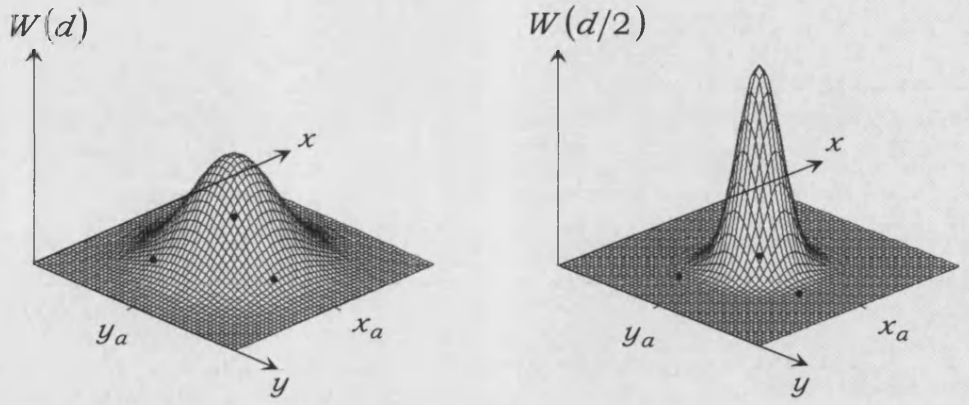


Figure 2.1. SPH Gaussian kernel centred on particle *a*. Black dots are randomly distributed neighbouring particles. The kernel tends to a Dirac distribution as *d* tends to zero.

fluid distortions, the transfer of momentum between cells and particles introduces diffusion into the flow, and the use of a limited number of particles is prone to introducing discontinuities in the pressure and density fields. Typically, work was carried out in developing more accurate and stable interpolations for the momentum transfers, ranging from the Cloud-in-Cell zero-order scheme of Birdsall & Fuss (1969), which sacrificed accuracy for stability and ease of implementation, to the higher order interpolations that led to a smooth distribution of conserved quantities and the SPH method.

2.1.2. Eulerian Methods

Another strategy to improve the PIC method is the incompressible Marker-and-Cell (MAC) scheme of Harlow & Welch (1965). Contrary to the PIC code, in the MAC method the particles are not used in the calculation (and are thus called markers) which is restricted to the underlying grid, making the method Eulerian in character. The markers simply show which cells contain fluid and, moreover, which cells lie along the free surface. From a known velocity field, the pressure field is computed from a Poisson equation and this yields local accelerations and the markers are then advanced according to neighbouring accelerations. Problems arise in the conservative transfer of

material quantities when markers enter a previously empty cell. Also, special provisions must be made when high flow velocities separate the markers sufficiently to empty an interior grid cell of markers. Additionally, if surface tension is to be included, the orientation of the free surface must be known and the determination of this slope from the distribution of markers is prone to considerable errors. Finally, the computational costs of keeping and moving a large number of marker particles in addition to the underlying Eulerian grid can be considerable. Amsden & Harlow (1970) propose the simplified marker-and-cell method where a predictor-corrector algorithm is used to solve the Poisson equation for the pressure. Miyata (1986) employs the same solution procedure in the TUMMAC (Tokyo University Modified Marker-and Cell) code for two-dimensional viscous incompressible flow, but here the free surface is modelled using an ordered sequence of discrete line segments connected to grid lines. The use of line segments is advantageous since these can be used as pseudo-vectors to determine whether grid nodes are located inside the fluid or not. This is a technique employed in the work described herein. The method requires velocities to be extrapolated to cells outside the fluid domain so that the free surface velocities can then be computed using linear interpolation from neighbouring values, and subsequently moved in the Lagrangian fashion of MAC methods. The TUMMAC code is able to model breaking waves with limited surface reconnection, although the problem of dealing with the entrainment of air is considerably simplified. The zero- and first-order calculations of the surface velocities seem to have a considerable negative effect on the accuracy of the solutions. Furthermore, the surface reconnection routine seems limited to simple geometrical configurations. Miyata & Yamada (1992) extend this algorithm to deal with three-dimensional flows.

Chen *et al.* (1991) improve the efficiency of SMAC by employing a preconditioned conjugate gradient method, and by restricting cell flagging operations to cells on or near the free surface. More significantly, the interior markers are eliminated and the location of the fluid field is determined using a single string of markers stretching along the free surface profile, though these are not connected by line segments as in the TUMMAC method. Chen *et al.* (1995) propose a new way of implementing the free surface boundary conditions which ensures symmetry of the solution of symmetrical problems. Recently, Park *et al.* (1999) have employed a modified MAC method in a finite difference scheme to investigate the interactions of non-linear waves with stationary three-dimensional bodies inside numerical wave tanks.

Other Eulerian schemes have been developed since the MAC scheme. Nichols *et al.* (1981) and Hirt & Nichols (1981) propose a volume-tracking algorithm which dispenses with the volume-tracking markers of the MAC method, and therefore is computationally more efficient. In the case of two fluids, the VOF method simulates the flow of both fluids, considered as one single fluid whose physical properties vary across the interface. The physical characteristics (density ? and the viscosity μ) of the fluid are determined by an additional function c, the volume fraction. The value of c is such that, if a cell is full of fluid 1 $c=1$ while if it is full of fluid 2 $c=0$. If it contains a mixture of the two fluids, i.e. if it lies on the interface between the fluids, it will have a fractional value between 0 and 1. The properties of the fluid are thus determined by,

$$\begin{aligned} \rho &= c\rho_1 + (1-c)\rho_2 \\ \mu &= c\mu_1 + (1-c)\mu_2 \end{aligned}, \qquad (2.3)$$

The time dependence of function c is governed by a scalar transport equation, which, for incompressible flow, takes the form

$$\frac{\partial c}{\partial t}+\nabla(\mathbf{u}c)=0, \tag{2.4}$$

where $\mathbf{u}$ is the velocity field and t is time. The discontinuity in c across the interface is approximated by a straight line determined from the value of c at each surface cell and its gradient. Derivatives of c are computed from neighbouring values but care must be taken due to its step nature. The accuracy of the VOF method also depends on the ability to advect the volume fraction function without diffusion and consequent smearing of the interface. The accuracy of the advection in turn depends on the accurate reconstruction of the interface. Hirt & Nichols (1981) propose an upwind donor-acceptor method. Although the formulation is intrinsically conservative, the method suffers from some diffusion nonetheless and consequently does not conserve volume. Values of c occasionally drift outside the prescribed limits, or become too close to 0 and 1, requiring that some adjustments be carried out, causing smearing of the interface. By imposing symmetry between the fluid phases, Lafaurie *et al.* (1994) eliminate the need for the bookkeeping operations aforementioned and recover exact volume conservation in their code SURFER. To achieve this, they use a mixture of upwind and downwind fluxing schemes depending on the angle between the interface and the flow. Special provision is made for the flotsam that results from the advection of the front so that this is also fluxed and not deleted as in the VOF method. Additionally, the appearance of flotsam is greatly reduced by the application of a small surface tension. To cope with the step discontinuity in c at the interface, Lafaurie *et al.* (1994) construct a smoothly varying volume fraction by means of a smooth interpolation kernel initially proposed by Brackbill *et al.* (1992) in their

continuum surface force (CSF) model. This allows an accurate computation of the surface normal and correct application of small values of surface tension. For higher values, however, the authors report the emergence of parasite currents. Considerable work has been carried out towards developing less diffusive bounded differencing schemes for the discretisation of the volume fraction function. Ubbink (1997) proposes that, rather than directing efforts towards investigating the criteria as to when to change between upwind and downwind fluxing, the downwind scheme should instead be replaced by higher resolution schemes. He applies a version of the high resolution QUICK scheme of Leonard (1979) for arbitrary meshes to obtain the CICSAM (Compressive Interface Capturing Scheme for Arbitrary Meshes) scheme. Jeong & Yang (1998) use the VOF method with a finite element scheme on adaptive quadtree grids. They use the filling pattern technique to determine the shape of the interface, by considering eight possible configurations of the values of function c in the neighbourhood of a cell. Chen *et al.* (1997) uses a modified VOF method to model three-dimensional gas bubbles rising in a cylinder of liquid, including bursting through the free surface.

The Volume-Of-Fluid method offers the advantages of being capable of dealing with complex flow problems whilst keeping computational costs low. Its treatment of two-phase flows also allows an easier analysis of such phenomena as air entrapment when compared with particle methods. However, the problem of maintaining a defined interface, of importance in wave studies, requires further attention. In the past, the problem of reconstructing the interface from known values of cell volume fraction has been addressed by several studies. The Simple Line Interface Calculation (SLIC) method of Noh & Woodward (1976) splits the advection process in each of the Cartesian directions and thus approximates the interface shape by

straight lines in the direction of the flux, as depicted in Figure 2.2b. The representation of the surface is thus changed depending on whether the x or y fluxes are being advected. Chorin (1980) improves the SLIC representation by considering both directions simultaneously, as shown in Figure 2.2c. Neither of these techniques, however, provides surface orientation, paramount, not only in the accurate application of boundary conditions, but also in the choice of discretisation of the convective transport equation for the volume fraction function. Toward this end, Youngs (1982) uses the neighbouring values of the volume fraction function to estimate both the position and the orientation of the surface, so that the reconstruction of the interface is that shown in Figure 2.2d.

Another way of cataloguing moving boundary methods, other than the Lagrangian-Eulerian divide used so far, is to distinguish them by the way they treat the interface. In this manner, one could describe methods that treat the interface explicitly either by placing Lagrangian particles along it or by using a mesh fitted to the boundary, as *surface tracking* methods. The LINC, ALE, CAVEAT and FLAG codes are illustrations of this with the work of Chern *et al.* (1986) and Unverdi & Tryggvason (1992) being other notable examples. Their major advantage is that the shape and position of the interface remains sharply defined throughout the simulation. Their drawback is their complexity which limits the range of surface phenomena that can be modelled. On the other hand, methods that use some measure of the volume of the fluid to reconstruct the free surface, by either employing a scalar function for the volume fraction like the VOF method, or the distribution of a set of marker particles as in the MAC approach, may be called *volume tracking* methods and are Eulerian in nature. These methods do not track the surface position explicitly but rather capture it from the values of fluid volume at each cell.

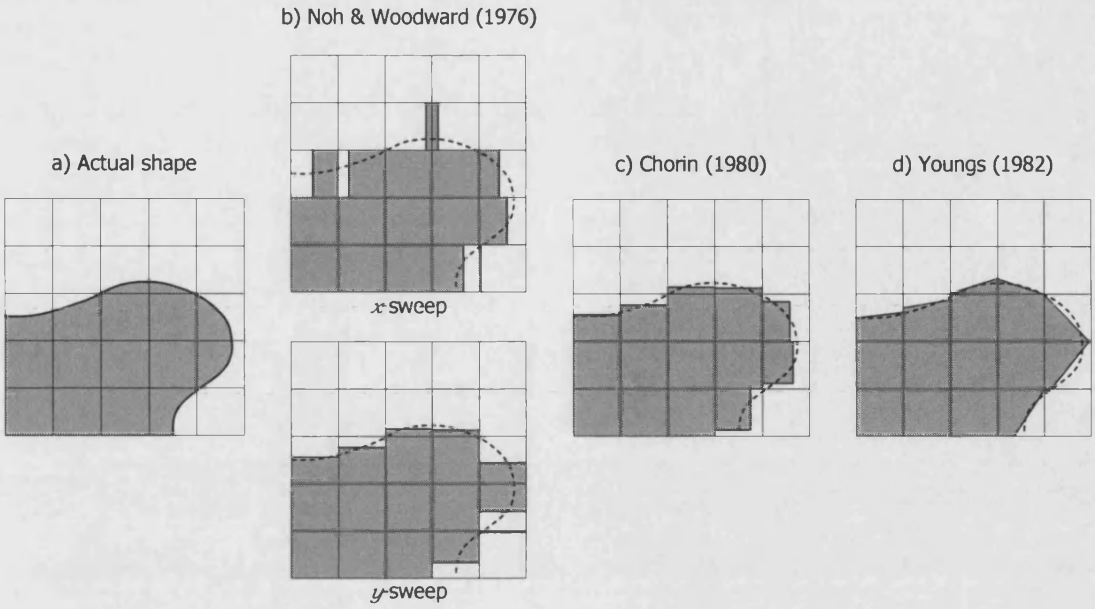


Figure 2.2. Actual (a) and reconstructed (b, c, d) shapes of a sample interface for different volume tracking methods.

They are generally simpler to implement and can model a broader range of flow features. However, their reduced definition of the interface makes them less preferable for certain studies where the interface position is paramount.

A method that falls somewhat in between these two definitions is the level-set method proposed by Osher & Sethian (1988) and Sussman *et al.* (1994). They suggest the use of a continuous function ψ, defined at each point in the domain as the shortest distance to the interface, positive for one fluid and negative for the other. It follows that the interface will be the zero level set of function ψ. For incompressible flow, the zero level-set of function ψ is moved according to the scalar transport equation

$$\frac{\partial \psi}{\partial t} + \nabla(\mathbf{u}\psi) = 0 , \tag{2.5}$$

in the same way as the volume fraction function of the VOF method is moved using equation (2.4). However, since ψ is a smooth Lipschitz continuous function by virtue of how it is defined, equation (2.5) is more easily solved

numerically for ψ than would be the case for the density or viscosity. Osher & Sethian (1988) have shown that, after ψ is convected, the interface remains identified by the level-set $\psi=0$. However, in the remainder of the domain ψ ceases to be a distance function and must be reinitialised as such, otherwise it can become irregular and the solution deteriorates severely. Sussman *et al.* (1994) point out that the efficiency of the method lies in this process of reinitialising ψ. To find the level-set contour and then determine the shortest distance of each grid node to it incurs severe computational costs. They therefore propose a different technique that dispenses with the need of locating the interface at each time step. It involves substituting the solution of equation (2.5) into

$$\frac{\partial \psi}{\partial \tau} = \text{sign}(\psi)(1-|\nabla \psi|) \tag{2.6}$$

and evolving it in the pseudo-time τ until steady state is reached. The sign function is smoothed across the interface using,

$$\text{sign}(\psi) = \frac{\psi}{\sqrt{\psi^2+\varepsilon^2}}, \tag{2.7}$$

where ε is a small parameter controlling the range of smoothing which is in essence the thickness of the interface. This remains constant in time. Equation (2.6) has the property that at the interface the zero level-set of ψ remains intact, since the sign function will be zero when ψ is zero, whilst in the rest of the domain it will converge to $|\nabla \psi|=1$, that is to the actual distance. Sussman *et al.* (1994) solve equation (2.6) on a regular Eulerian mesh using an Essential Non-Oscillating (ENO) method for the spatial derivatives and present results for flows with surface tension including the rising of an air

bubble in water, a water drop in air and the merging of water drops. The ENO scheme was found to introduce some errors near the interface cells, which Russo & Smereka (2000) propose to reduce using a modified formulation of equation (2.7) at the interface cells. Colicchio *et al.* (2002) modify the smoothing function further and present results for the case of a surface piercing vertical flat plate travelling at known velocity in a water tank. They report that numerical diffusion still occurs across the cells at the interface.

The level-set method is purely Eulerian and most often employed on a fixed grid. It differs from the other Eulerian methods described here in that it does not track volume as the VOF and MAC schemes do. It tracks the surface instead but without recourse to Lagrangian particles or grids. It does so implicitly and as a result has potentially great versatility and robustness. However, it is also plagued by problems of diffusion at the interface that higher order advection schemes have so far failed to eliminate.

2.1.3. Boundary Methods for Irrotational Water Waves

The previous discussion of the current research into methods for flows with moving boundaries was not restricted to any particular type of flow. In most of the literature cited they are employed in the solution of the Navier-Stokes equations for either compressible or incompressible flows with or without surface tension. However, certain fluid flows may be accurately described by a less encompassing mathematical model. Water waves, for example, may be modelled by much simpler theory by disregarding incompressibility, viscosity and surface tension and considering the flow to be irrotational. The criteria under which these assumptions are valid will be debated in Chapter 4.

When the mathematical model is thus simplified the numerical method required to solve the governing equations may also be considerably simpler. Irrotational surface waves are described by a Laplace equation for the velocity potential, $\nabla^2\phi=0$, and dynamic and kinematic boundary conditions at the free surface. Fluid velocities are given by the gradient of ϕ. The problem is fully non-linear and has no exact analytical solution. Within a closed boundary, however, the velocity potential is uniquely determined by its values on the boundary itself. This leads to the boundary element method, which was first applied by Longuet-Higgins & Cokelet (1976) to solve the fully non-linear problem. Their mixed Eulerian-Lagrangian (MEL) method has two steps: at any instant of time, a boundary-integral equation is solved for the gradient of the velocity potential normal to the surface using an Eulerian frame of free surface elements, after which Lagrangian points on the free surface are moved and have their value of the velocity potential updated. The wave motion is assumed to be periodic in space and the water is taken to be of sufficient depth so that the slow diffusion of vorticity inwards from the boundaries can be neglected (see Longuet-Higgins (1953)). These assumptions remove the exterior boundaries and, in their stead, periodic boundary conditions are used to achieve far-field closure. By imposing an asymmetric distribution of pressure on the free surface the MEL method can simulate limited overturning of the wave crest.

BOUNDARY CONDITIONS

The boundary integral approach has the fundamental advantage that the velocity potential need only be known at the boundaries. With the assumption of spatial periodicity and infinite depth, Longuet-Higgins & Cokelet (1976) in fact restricted the calculations to the free surface alone. This

implies that the resulting matrices are one order smaller (N) when compared with methods that discretise the whole domain (N^2). It should be noted, however, that these matrices are fully populated and are less amenable to fast matrix inversion algorithms.

Following the pioneering boundary integral MEL method, several studies followed. Faltinsen (1977) used it to investigate the heaving motion of a two-dimensional floating body. Rigid bodies are treated by introducing Neumann boundary conditions prescribing that the fluid velocities normal to the bodies be equal to the normal velocities of the body surface, f_b,

$$\frac{\partial \phi}{\partial n} = f_b . \tag{2.8}$$

With a mixed set of boundary conditions, the integral representation of MEL methods leads to a second kind Fredholm integral equation for nodes on rigid boundaries where a Neumann condition is applied, and to a first kind integral equation for nodes on the free surface where ϕ is known. Solution of the resulting boundary value problem yields $\partial\phi/\partial n$ at the surface and ϕ at rigid boundaries. This then permits the application of the kinematic and dynamic boundary conditions at the free surface and the advancement of the solution by numerical integration in time. When dealing with free body motions, however, the dynamic equations of the body motion and the fluid problem must be solved simultaneously as each depends on the other. This requires the hydrodynamic force on the body to be known at each time step, which in turn calls for knowledge of the time derivative of the potential, $\partial\phi/\partial t$. This difficulty is usually overcome by solving an auxiliary boundary value problem for $\partial\phi/\partial t$, with Neumann conditions on the body surface expressed in terms of the velocity and accelerations of the body (Vinje & Brevig (1981),

Cointe *et al.* (1991), Wu & Eatock Taylor (1996)). Since $\partial\phi/\partial t$, like ϕ, also satisfies Laplace equation and its value on the free surface is easily deduced from the dynamic boundary condition on the free surface, the kernel of the discretised problem is the same. Tanizawa (1995) reformulated the potential flow problem in terms of Prandtl's non-linear acceleration potential, $\hat{\phi}$, defined as,

$$\hat{\phi} = \frac{\partial \phi}{\partial t} + \frac{1}{2}(\nabla\phi)^2, \tag{2.9}$$

and showed that the fluid acceleration, $\mathbf{a}$, is given by the gradient of the acceleration potential, $\mathbf{a} = \nabla\hat{\phi}$. Since the non-linear acceleration potential is equal to the hydrodynamic pressure, the free surface dynamic boundary condition is simply $\hat{\phi} = p_a$, where p_a is the atmospheric pressure, which may be set to zero without affecting the acceleration field. Boundary conditions on body surfaces are given by,

$$\frac{\partial \hat{\phi}}{\partial n} = \mathbf{n}\cdot\boldsymbol{\alpha} + q, \tag{2.10}$$

where $\mathbf{n}$ is the generalized normal vector to the body surface, $\boldsymbol{\alpha}$ is the generalized acceleration vector of the body and q is the term due to the fluid velocity which Tanizawa expresses in terms of the velocity potential and the angular and translational velocities of the body. The acceleration of the body $\boldsymbol{\alpha}$ is obtained from Euler's equation of solid body motion coupled with fluid motion which yields an implicit boundary condition expressing the relation between the acceleration potential and its normal on the body surface. Tanizawa (1995) employed this formulation to the study of three-dimensional motions of floating bodies. Tanizawa (1996) extended this formulation to deal

with multiple fluid domains, and presents results for floating ships with partially filled tanks.

When the domain is not bounded by rigid boundaries, far-field closure must be satisfied. The aforementioned studies employed the assumption of spatial periodicity to achieve this. However, due to the presence of a floating body, Faltinsen (1977) cannot accurately presume the motion to be periodic. Instead, the non-linear inner solution is matched to that of a Rankine dipole in the far field. The discrepancies between the two flows, however, require the computational domain to be continually increased as a function of simulation time and the ensuing computational costs restrict results to less than an oscillation period. The situation is even more critical for three-dimensional studies, as exemplified by Isaacson's (1982) investigation of non-linear diffraction by a vertical cylinder, where the fluid velocities were assumed to be zero on a finite truncation boundary. Vinje & Brevig (1981) extended the approach of Longuet-Higgins & Cokelet to include finite water depth and floating bodies but retained the assumption of spatial periodicity by employing a sufficiently large domain and thus assuming that the local field around the floating bodies is largely unaffected by the periodic boundary conditions; in particular, Greenhow *et al.* (1982) employed the same technique to study capsizing of a duck wave energy device in extreme waves. Dommermuth & Yue (1987) solved three-dimensional axisymmetric problems using the MEL method, namely the axisymmetric heaving of a floating vertical cylinder. The formulation employed Rankine ring sources to implement boundary conditions at the body surface, as well as the horizontal bottom of the tank. Far-field closure was achieved by matching at the cylinder surface the non-linear computational solution to a general linear solution of transient outgoing radiated waves.

SMOOTHING OF FREE SURFACE VARIABLES

In their computations, Longuet-Higgins & Cokelet (1976) observe that the wave profile develops a saw-toothed appearance after some time during the simulation. Longuet-Higgins & Cokelet speculate that the cause of this may be a physical phenomenon that in an actual fluid is dampened by viscosity and smoothed by surface tension. These instabilities in the free surface profile, however, cause numerical problems in the computation of spatial derivatives. They advocate the use of Chebyshev polynomials (see Section 4.6) to smooth the co-ordinates and velocity potential of the Lagrangian particles, although some loss of wave energy is observed as a result. The short wavelength instabilities in the surface profile have been confirmed by many an author since then and smoothing polynomials have often been employed (see for example Dold (1992), Wu & Eatock Taylor (1994) or Turnbull (1999)).

Contrary to the assertions of Longuet-Higgins & Cokelet (1976), Dommermuth & Yue (1987) believe the short wavelength instabilities of the free surface profile to be non-physical, attributing them instead to inaccuracies in the calculation of the velocity of free surface particles; specifically, they consider that the cause may be narrowed down to the concentration of the Lagrangian particles in regions of higher gradients of the velocity potential, so that the Courant stability condition is inevitably violated. With this in mind, they propose a regridding of the Lagrangian surface particles after each time step, in order to maintain an evenly spaced distribution. Although the procedure has in itself a smoothing character, they find that this approach has less severe effects on the conservation of the wave energy, and results in a more robust code.

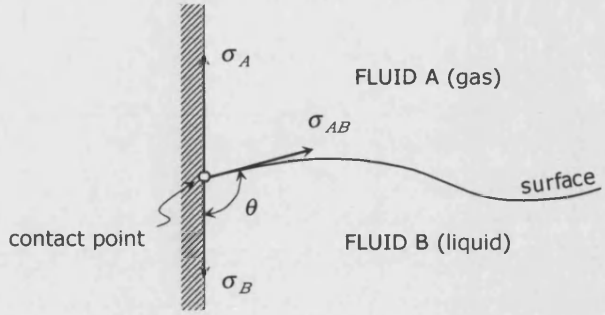


Figure 2.3 Contact point at the intersection of free surface with rigid boundary

INTERSECTION OF FREE SURFACE AND RIGID BOUNDARIES

An important consideration in MEL methods is the confluence of boundary conditions at the intersection between the free surface and rigid bodies: at the free surface a Dirichlet condition is prescribed whilst at the rigid wall a Neumann no-penetration condition stipulates the normal gradient of the potential to be zero. This difficulty reflects the complexity of the real case (i.e. without any of the assumptions of irrotational inviscid flow). The weak singularity in the solution at the contact point, due to a discontinuous stress tensor caused by the three surface tensions between the three phases (gas-solid σ_A, liquid-solid σ_B and gas-liquid σ_{AB}, as depicted in Figure 2.3) is not fully understood. The problem was initially studied in the field of capillary flow where it has a much more influential bearing on the physics of the flow. Dussan (1979) provides an early discussion of theoretical and experimental work. Attempts to fully understand the physics of the three-phase contact have led to the treatment of the contact point as a region of finite length, rather than an absolute step discontinuity, and molecular models have been developed to predict values for the contact angle, θ (Saville (1977)).

At the much larger length scale of MEL methods for irrotational flows, the singularity at the contact point has a global influence in the whole

computational domain and can adversely affect the solution. The singularity must be resolved by the imposition of both boundary conditions simultaneously. For a vertical two-dimensional piston moving in the x-direction in a mean water depth, h, Peregrine (1972) derived an analytical solution for the perturbation of the surface profile whereby it displayed a $t\log(\tanh(\pi x/4h))$ behaviour for small time t, with higher order terms having been neglected. This result has been confirmed by a number of other investigations both numerical and experimental (for example Lin *et al.* (1984) and Wu & Eatock Taylor (1994) in two-dimensions, and Wu *et al.* (1997) in three dimensions). For two-dimensional problems, Lin *et al.* (1984) specify both the stream function and the velocity potential at the intersection in a Cauchy-integral solution, and report no computational difficulties. Cointe *et al.* (1991) employ a numerical treatment at the surface-body intersections based on a local asymptotic analysis in the weakly non-linear regime that corresponds to a small vertical acceleration of the body. They refer the reader to Cointe (1989) for more details on this procedure.

Dommermuth *et al.* (1988) combine the Cauchy-integral formulation of Vinje & Brevig (1981), the treatment of the surface-body intersection of Lin *et al.* (1984) and the regridding technique of Dommermuth & Yue (1987) to produce high-resolution computations of breaking waves that compare well with their experimental data and thus give a good indication of the validity of using potential theory for describing gravity waves.

Zhou & Gu (1991) propose a three-dimensional boundary element method to study non-linear wave radiation and diffraction around structures and floating bodies. Despite conceding that the procedure lacks physical justification, difficulties associated with the singularity at the intersection of

surface and body force the authors to resolve them by interpolating on the free surface.

Mercer & Roberts (1992) employ a boundary element method to study limiting profiles of standing waves using the crest acceleration as the determining factor and to investigate their stability using perturbations in the harmonic and sub harmonic ranges. They found standing waves to be stable to harmonic perturbations and always unstable to some sub harmonic mode that is dependent on the wave steepness.

Ferrant (1997) employs the MEL code ANSWAVE to study three-dimensional applications. The surface is represented by a series of triangular Rankine panels defined by a distribution of surface nodes. Semi-Lagrangian and fully Lagrangian descriptions of the free surface are used in turn to study wave diffraction from vertical cylinders and the behaviour of large amplitude standing waves in a rectangular tank. Using the explicit model for the non-linear incident wave of Rienecker & Fenton (1981), the flow is split into incident and perturbation contributions, offering substantial savings in computational time. Bi-cubic splines are used to represent both the velocity potential and the vertical co-ordinates of the Lagrangian surface particles. Differentiation of the spline formulae yields the normal vector and velocity to the surface. The polynomial-based smoothing of Longuet-Higgins & Cokelet (1976) is used to remove saw-tooth instabilities.

2.1.4. Full Domain Methods for Irrotational Water Waves

As demonstrated by the number of citations in the previous section, the boundary element method has been extensively used in the solution of the non-linear problem in the time domain although success of this technique for the full three-dimensional analysis of a practical structure is as yet limited.

Several difficulties exist in this method, though, some of which have already been mentioned. The confluence of boundary conditions at the free surface and rigid boundaries causes a singularity in the boundary element or panel solution that often precipitates divergence. Moreover, boundary element methods are restricted to potential flow, since addition of viscosity (as Price & Tan (1992) did using a convolution integral formulation) implies that the whole fluid must be discretised and the main advantage of the method vanishes. Also, it is often more convenient to implement boundary conditions on bodies of complex geometry if the interior of the fluid domain, and not just its boundary, is discretised. Finally, although the coefficient matrix of boundary element methods is one order smaller than that of methods that discretise the whole domain, the fact that it is fully populated implies that every matrix element must be stored during computation. This implies that computer memory requirements for the boundary element method can sometimes exceed those of finite difference (FD) or finite element (FE) methods which usually produce diagonal matrices with reduced bandwidth demanding only a fraction of the storage. This is exemplified in Figure 2.4. Of course, storage of the dependent variables is lower for boundary schemes due to a smaller number of computational nodes. However, there are many cases in which methods that discretise the whole domain can be more economical than boundary integral methods (see Wu *et al.* (1997)). With these considerations in mind, Eatock Taylor & Wu (1986) employ a coupled finite element method to solve the hydrodynamic problem of an oscillating horizontal cylinder without forward speed, which Wu & Eatock Taylor (1987) extend to deal with oscillating cylinders with forward speed. The domain is divided into two regions. An inner region surrounding the cylinder is discretised using a finite element mesh whose outer boundary is matched to a boundary integral

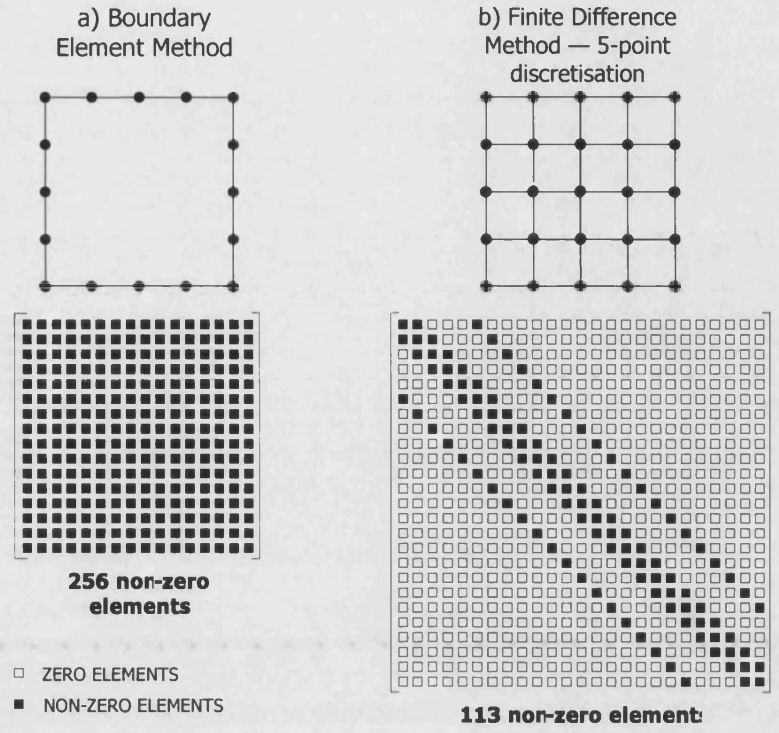


Figure 2.4. Example of memory storage requirements of the coefficient matrix for the boundary element and finite difference methods for a simple domain.

solution of the flow in the far-field region. The method has the particular advantage that the second derivatives of the velocity potential, present in the boundary condition at the cylinder surface due to the forward motion, need not be calculated when using the finite element formulation, and are instead replaced by first derivatives which are much more easily obtained by differentiating the finite element shape functions only once.

Wu & Eatock Taylor (1994) discretise the whole fluid domain using triangular finite elements to simulate a wave maker and standing waves in a container. Solutions are obtained by reformulating the field equation and boundary conditions into a variational statement and computing the velocities by the Galerkin method where the velocity field $\nabla\phi = \mathbf{u}$ is approximated using $\int_{\Omega} N_i(\nabla\phi - \mathbf{u})d\Omega$, where O is the fluid domain and N_i are the element shape functions. After each time-step a new mesh is regenerated and the potential

field solved using the updated boundary conditions. Turnbull (1999) modified this finite element formulation by employing a sigma variable transformation which eliminated the need for remeshing after each time step. Greaves *et al.* (1997) apply the same method to triangularized adaptive quadtree grids, using the triangulation method introduced by Greaves (1995).

Wu *et al.* (1997) use a three-dimensional finite element method with domain decomposition, and demonstrate the efficacy of the finite element scheme for computing irrotational flows around a vertical cylinder. In particular, they investigate the optimum amount of subdomain overlapping and the choice of relaxation factors.

Yeung & Ananthakrishnan (1989) use boundary-fitted meshes with a co-ordinate transformation procedure to study wave overturning using a finite difference formulation. Yeung & Vaidhanathan (1990) employ a finite difference scheme to simulate non-linear free surface potential flows over

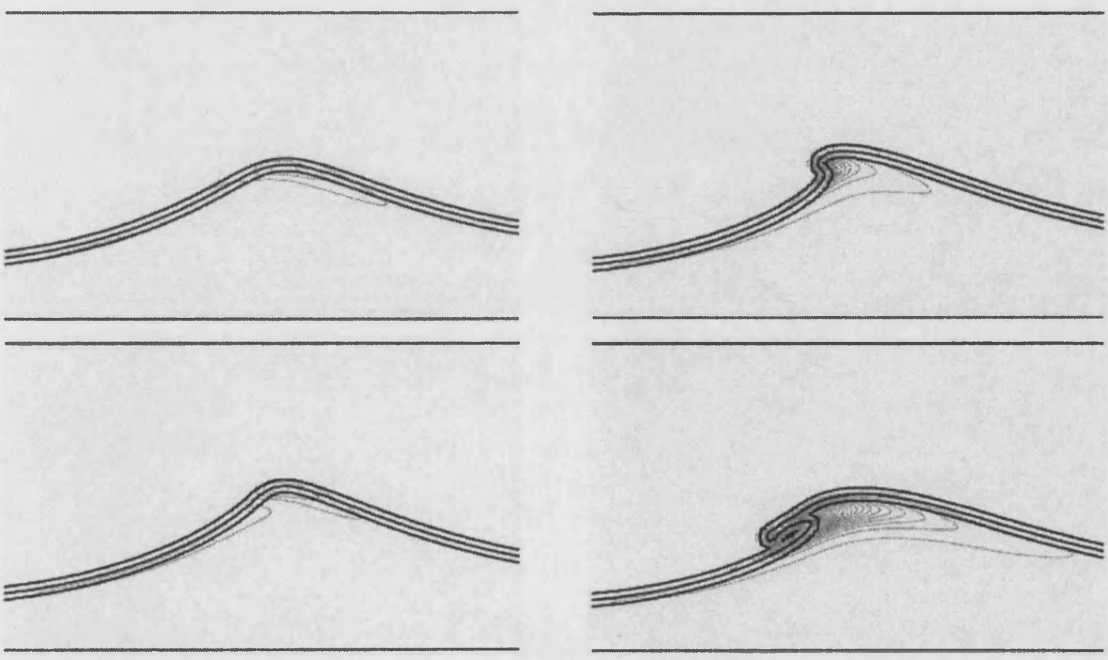


Figure 2.5. Evolution of the free surface and of the vorticity × density contours for the level-set solver of Iafrati & Campana (2003) for air-water flow. The thickness of the free surface can be seen from the spacing of three density levels (20, 500,980)

submerged circular and semi-circular cylinders.

An interesting combination of the type of methods described in this section and those in Section 2.1.2 is proposed by Iafrati & Campana (2003). The authors are able to model complex wave phenomena such as wave breaking by dividing the fluid domain into two subdomains. In the subdomain covering the neighbourhood of the surface, a water-air viscous flow model is used and the surface is tracked using a level-set technique, whilst a potential flow model is adopted to describe the irrotational flow in the water region deep under the free surface. Each subdomain is solved in turn to provide boundary conditions to the other subdomain solution. The authors investigate two coupling techniques: one uses the normal stresses along the boundary separating the domains to supply a pressure field to close the viscous solution; the other technique uses an overlapping region and a mixture of Neumann and Dirichlet conditions. Figure 2.5 illustrates the wave breaking history they produced and shows the discrete thickness of the interface, characteristic of level-set methods.

2.2. Hierarchical Cartesian Grids

The choice of discretisation grid is strongly dependent on the type of problem being modelled. Two considerations are often paramount: the need for localised refinement in the areas of the computational domain where the gradients of the independent variables are higher; and accurate boundary fitting which affects the accuracy with which boundary conditions are implemented. There is a variety of fully unstructured grid generators that can tackle both these issues successfully (see for example, Fletcher (1991) for a review). However, fully unstructured meshes are expensive in terms of computational time, storage and complexity of the algorithms.

Hierarchical Cartesian grids offer variable refinement with only a small increase in computational costs, since they retain a structured nature that greatly facilitates discretisation and grid manipulation procedures. They are constructed by dividing cells into smaller ones, called children (Figure 2.6a). This generation procedure produces a tree-like grid structure (Figure 2.6b) which has useful properties: a) they provide variable refinement whilst preserving an ordered structure; b) their tree-like structure requires limited memory costs whilst allowing fast neighbour finding routines; c) they are fast to generate; and d) their stratified structure makes them amenable to the use of multigrid techniques. In two dimensions, quadrilateral hierarchical Cartesian grids are also called quadtrees, whilst in three dimensions the name octree is often used.

The main drawback of this type of mesh is their poor fitting to curvilinear geometries which derives from the quadrilateral (or hexahedral in three dimensions) topology of their cells. This may be improved by triangulating the cells (see, for example, Greaves *et al.* (1997)), or by cell-cutting techniques as used here. An additional disadvantage of quadtrees is the occurrence of hanging nodes — those that lie along the face of a cell —,

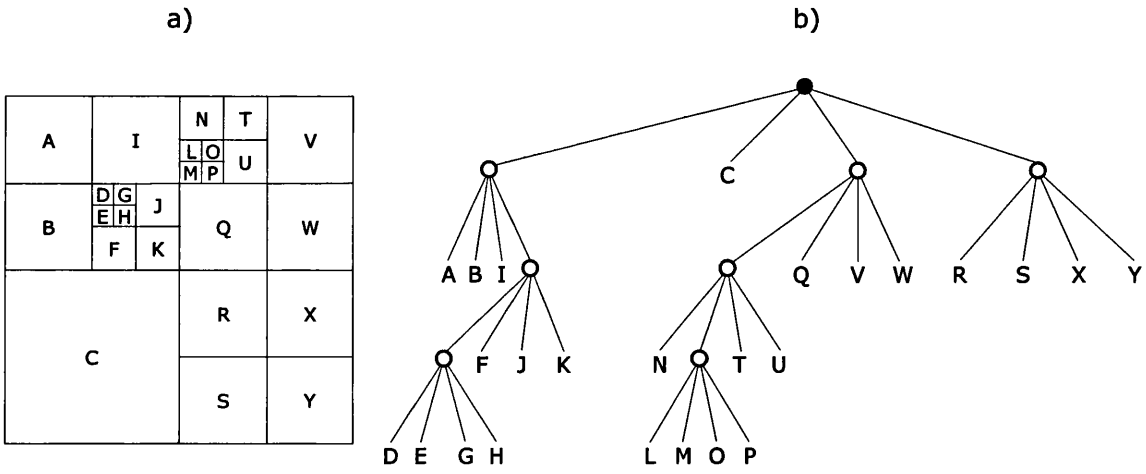


Figure 2.6 Sample quadtree grid (a) and its associated tree-like structure (b)

since they require special treatment when any vertex-centred discretisation scheme is used.

This type of grid was first developed in the field of computational image processing, but has since found uses in diverse areas of science. Samet (1990) discusses several applications in the field of computer graphics, whilst in the field of computational fluid dynamics, Schmidt (1991) and Evans (1993) use them in the study of compressible flows. De Zeeuw & Powell (1993) employ them in conjunction with a Riemann solver to solve the steady Euler equations; and Young *et al.* (1991) make use of quadtrees to investigate aerodynamic potential flows. Jeong & Yang (1999) use adaptive octrees in a three-dimensional finite element code to solve free surface flows and present results for the breaking of a three-dimensional dam.

One of the important parameters when managing any numerical grid is the choice of how much grid information to store and how to store it. This choice must be balanced with the computational costs that ensue when data that is not kept in memory must be calculated during computation. Quadtree grids offer a few possibilities in the way of data handling. In his discussion of

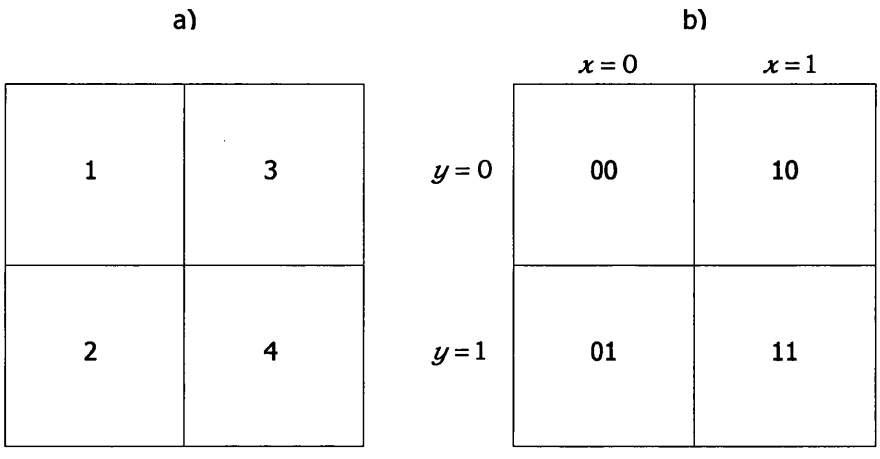


Figure 2.7. Numbering of quadrants in quadtree meshes: a) Samet (1982); and b) van Dommelen & Rundensteiner (1989)

neighbour finding routines where he introduces fundamental concepts such as mirror paths, Samet (1982) stores grid information as a single reference number, n, for each cell. This is obtained using,

$$n = n_p + 5^l i \tag{2.11}$$

where n_p is the reference number of the parent cell, l is the division level of the cell and $i \in \{1,2,3,4\}$ is the quadrant position of the cell according to the convention of Figure 2.7a. From this integer data, size, co-ordinates and neighbourhood of a cell can be determined. In the present work, a variant of this numbering method was designed and the details of its implementation are discussed at length in Chapter 3.

In their vortex tracking calculations, van Dommelen & Rundensteiner (1989) propose a different storage method that uses a binary representation of the quadrants of a cell as shown in Figure 2.7b, made up of x and y binary co-

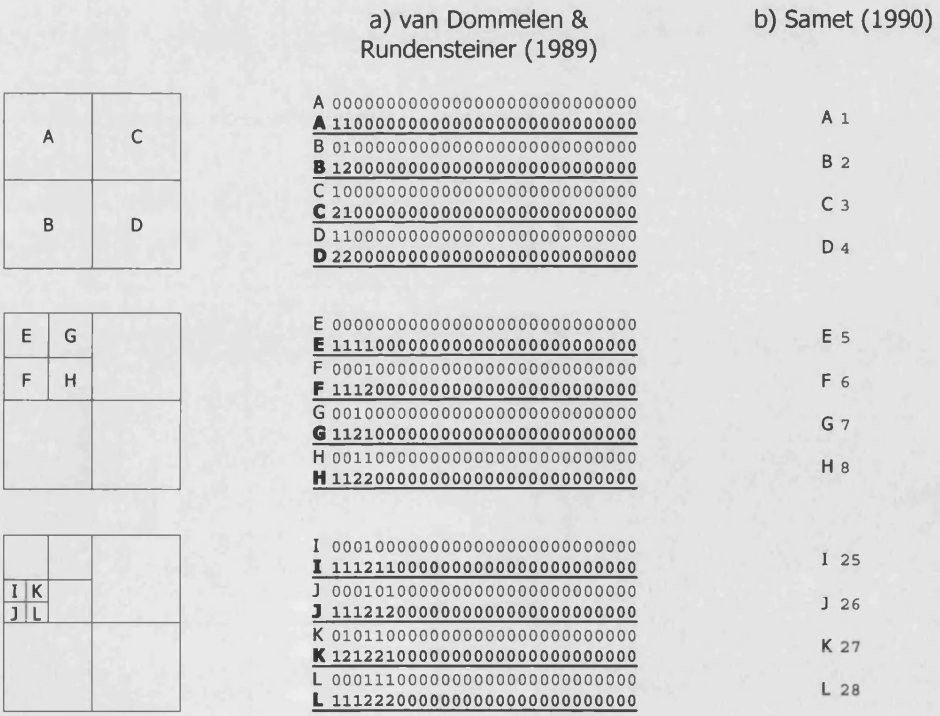


Figure 2.8. Two reference numbering systems for quadtree grids: a) van Dommelen & Rundensteiner (1989); and b) Samet (1990)

ordinates. The reference numbers of each cell are composed by concatenating the binary quadrant positions of the cell's ancestors. It can be seen that without modification, this system can yield identical reference numbers for different cells as exemplified in Figure 2.8a for cells A and E (reference numbers in normal face type). The problem arises from the fact that the trailing zeros present in the field may themselves be interpreted as binary representations of quadrant positions. To avoid this, van Dommelen & Rundensteiner (1989) added +1 to each binary digit, resulting in the base 3 system shown in underlined bold face type in Figure 2.8a.

A great disadvantage of this approach is the tremendous increase in memory storage that this simple addition of 1 entails. In principle, it would be possible to store the whole binary reference number as, say, a 32-bit word with an additional small integer to store the division level of the cell, instead of the +1 addition. The manipulations required for neighbour finding procedures and other grid operations could then be performed using fast bit operations.

Table 2.1. Memory requirements for two quadtree numbering systems

Maximum division level	Memory required (bytes per cell)	
	Samet (1982)	van Dommelen & Rundensteiner (1989)*
5	**2**	**3**
6	**2**	**3**
7	**2**	**3**
8	**4**	**3**
9	**4**	**4**
10	**4**	**4**
11	**4**	**4**
12	**4**	**4**
13	**4**	**5**
14	**4**	**5**
15	**4**	**5**
16	**8**	**5**
17	**8**	**6**
18	**8**	**6**
19	**8**	**6**
20	**8**	**6**

* These values were found from the minimum number of bytes that can contain the 'bits' required for each division level. An additional byte containing the division level of each cell is added.

However, these operations are generally specific to the combination of CPU and compiler used, and the portability of the code would be forsaken. This would be a very undesirable disadvantage for a computer code.

Even with a truly binary representation (i.e. without the addition of +1 to the binary position), the van Dommelen & Rundensteiner method is only more economical than Samet's for large quadtrees, that is, for quadtrees with a very high level of refinement. This is shown in the Table 2.1. On the other hand, it is computationally faster to extract grid data from the binary system because it does not require as many arithmetic operations as the integer-based Samet system. However, most techniques to recover information from Samet reference numbers are amenable to efficient recursive programming and this disadvantage is not severe.

An additional numbering system based on finite difference type-referencing is discussed by Yiu *et al.* (1996). It records three integer values for each cell: one that yields the x-path through the grid, another the y-path and the third the division level of the cell. As an aside, the Samet system can be interpreted as the amalgamation of these three sets of data.

2.3. Multigrid Iterations

The discretisation of the governing equations and boundary conditions, commonly yields a large algebraic system of equations which must be solved. The choice of solution method depends on the structure of the coefficient matrix of this system. Whilst direct methods, such as Gauss elimination or LU decomposition, can provide a more numerically accurate solution, the computational storage costs they incur are often prohibitive, unless the matrix has some special structure such as being tri-diagonal. As a result, iterative methods are commonly used to solve large systems, by which an initial guess

is successively corrected using the algebraic system until the correction becomes negligible.

The multigrid (or multilevel) idea was first suggested by Fedorenko (1964), who observed that relaxation (iterative) solvers are efficient at removing the high frequency components of the error, but suffer from slow convergence rates when reducing the low frequency smooth error components. Brandt (1977) first applied it to the solution of practical problems, namely to transonic flow calculations. Briggs (1987) provides a useful introduction to the several components of a multigrid solver, whilst Hackbush (1985) and Wesseling (1992) are more detailed texts on the subject.

It will be shown that low-frequency error components on a given grid appear as high-frequency Fourier modes on a coarser grid and thus make the numerical solution amenable to fast convergence. The idea behind multigrid iterations is thus to employ several grids of different cell size to maintain the oscillation of the error field and therefore exploit the high-frequency smoothing of relaxation methods. This is achieved by transferring the residual field to a coarser grid (a process called *restriction*) and solving an equation for the convergence error or correction. The resulting error field is then transferred to the finer grid (a process called *prolongation*) and used to update the solution of the governing equation. Figure 2.9a shows the grid sequence for this technique. The increased convergence speed must offset the additional computational costs of computing the residuals, of generating the other grids, and of transferring information between them for the method to be justifiably employed.

When the data transfer between grids is restricted to the residual and convergence error fields, the method is called correction scheme (CS). For strongly nonlinear problems, such as those required in the solution of the Navier-Stokes equations, the full approximation scheme (FAS) is sometimes used, in particular when solving the elliptic Poisson equation for the pressure. It is so called because, instead of solving for corrections, one seeks approximations to the solution at each grid. The solution obtained on each grid in FAS is not, however, the solution that would be obtained were that grid used by itself but a smoothed version of the fine grid solution.

Another advantage of using coarser grids to accelerate the solution lies in the fact that boundary information is more quickly imparted to the interior of the domain when coarser grids are used. This is the more so, the more explicit the iterative solver used. As will be discussed in the following two chapters, numerical solvers for quadtree grids are mostly point-by-point solvers, such as the Jacobi and Gauss-Seidel methods, due to the unstructured nature of the resulting coefficient matrices. These methods consider each grid cell in turn and apply the corresponding discretisation

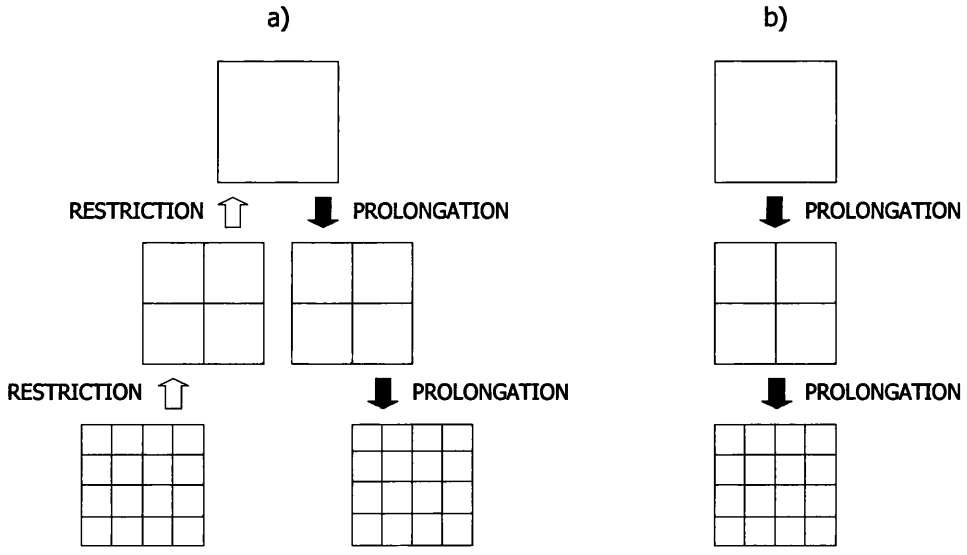


Figure 2.9 Multigrid strategies: a) correction scheme; b) initial guess

expression using the most recently updated values of the field variables at each of the cell's neighbours. It follows that, for an interior cell separated from a given boundary by n other cells, the boundary condition will only affect it after n iterations have been performed. However, if a grid twice as coarse is employed that number will be halved to $n/2$. Coarser grids are therefore often used to obtain a reasonably accurate initial guess on the fine grid. This is achieved by solving the governing equations at the coarsest grid level for a few iterations, and transferring the solution onto finer and finer grids until the finest grid level is reached. All calculations are performed on the governing equation and the correction equation is not used. The grid sequence is illustrated in Figure 2.9b.

In the field of computational fluid dynamics, multigrid solvers have been used extensively with a variety of grids. Darwish *et al.* (2003) use a multigrid strategy to accelerate a VOF type solver for multi-fluid flows of all speeds on irregular finite-volume quadrilateral meshes. The presence of a discontinuous function, such as the volume-fraction, poses interesting problems since prolongation transfer procedures tend to interpolate the data across the fine mesh in some way. This causes discontinuities to be smoothed, which in the case of a VOF solver, increases undesired smearing of the interfaces. Furthermore, Darwish *et al.* (2003) find that the use of a standard prolongation stencil can cause the volume-fraction function to fall outside its prescribed limits and they find that an additional iterative procedure is required to ensure the volume-fraction limits are respected. The smoothing properties of the prolongation operator and the numerical diffusion it entails make the multigrid method specially suited for elliptic problems, such as those described by Laplace or Poisson equations. Such is the case of Udaykumar *et al.* (2001), which employ the FAS multigrid technique to quickly

solve the Poisson pressure equation within a fractional step method. They use a fixed Cartesian grid that allows for the presence of complex submerged moving boundaries. The presence of boundaries of arbitrary shape inside the computational domain requires special treatment of the intergrid transfer procedures since the solid boundaries do not necessarily fall along grid lines. To avoid reconstruction of the immersed boundary at every coarse multigrid level, which can be a complex task, Udaykumar *et al.* (2001) use a volume-fraction approach to discretise the Poisson equation on the coarser grids, whilst retaining a sharp interface at the finest level.

In their code for the solution of the Reynolds-averaged Navier-Stokes (RANS) equations on unstructured grids, Hino & Hirata (2002) applied a multigrid strategy in an attempt to reduce the large computational costs associated with unstructured grid approaches. They study two cases involving free surface flows around container ships. In the first case, the free surface is treated as a symmetrical plane, and the multigrid technique reduces the number of iterations required by 70%. In the second case, a level-set technique is used to model the free surface geometry and the reduction in number of iterations provided by multigrid is only 40%. Values of CPU times are not provided. Hino & Hirata (2002) attribute the change in qualitative performance between the two cases to the way the free surface is treated, namely to the treatment of discontinuities across the interface.

The application of multigrid strategies to hierarchical Cartesian grids was pioneered by Gáspár & Józsa (1991) who use a cloud-in-cell method to solve the unsteady 2D Navier-Stokes equations in a finite-difference vorticity-stream function formulation. They create the coarse grid levels by successfully pruning the smaller branches of the quadtree and use linear

restriction and zero-order prolongation operators in initial guess and correction schemes. Gáspár *et al.* (1991) use an identical scheme to solve a transport equation for pure diffusion and convective diffusion problems, and Józsa & Gáspár (1992) study wind-driven flow patterns in shallow lakes. Berger *et al.* (2001) describe work on a multigrid solver on hierarchical Cartesian grids with domain decomposition for parallel computations. They employ a different grid coarsening scheme which yields a more uniform cell coarsening across the computational domain. These two different grid coarsening schemes will be discussed in more detail in Chapter 5.

3. Grid Generation

The review in Chapter 2 divided methods that model irrotational waves into two approaches: those that solve the governing equations along the boundaries of the computational domain, and those that discretise the entire domain. Whilst the former generally offers a higher computational efficiency, it is restricted to irrotational flows and is unable to model viscous phenomena. Even though the problems studied in this work are restricted to irrotational flows, it was considered desirable to develop a method which could be readily extended to other types of flows and, as a result, a full domain approach was chosen.

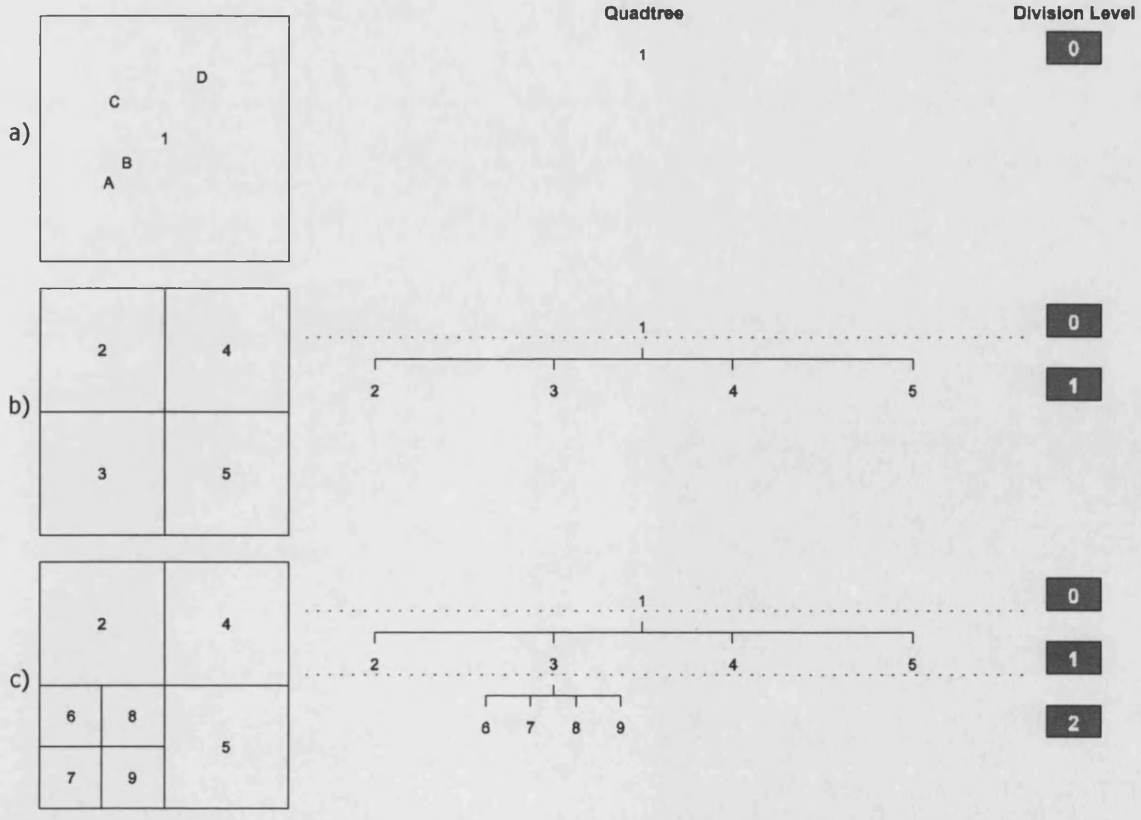
Grid generation is a crucial part of the process of obtaining a numerical solution to the physical problem in hand, as accuracy, speed of convergence and correct boundary condition implementation all depend upon the choice of grid. Quadtree grids were selected for this work for three main reasons. First and foremost, they provide variable refinement so that higher resolution may be used in areas of the domain where higher accuracy is crucial. Secondly, the tree structure of quadtree grids facilitates such tasks as neighbour finding, identifying a cell containing a set of co-ordinates, and discretisation of the governing equations. Finally, quadtree grids are ideally suited to the implementation of multigrid iterations as the generation of coarser grids is fast and straightforward.

This chapter will cover the grid generation algorithms and the techniques used to store and handle grid information.

3.1. Grid Generation Algorithm

The tree structure of quadtree grids creates certain hierarchical relationships between cells that instinctively suggest the use of some genealogical terms. These will be employed henceforth in the hope of simplifying the text. Though the significance of most terms will be self-evident, the following paragraphs will formally define this genealogical nomenclature.

Let us consider the unit square domain in Figure 3.1 as the root cell of the quadtree. Dividing the cell into four equal sized cells generates four branches stemming from the root cell, much like a genealogical diagram, as shown in Figure 3.1b. The newly created cells 2-5 are therefore the *children* of cell 1, and as a result are each other's *siblings*. Conversely, cell 1 is the *parent*



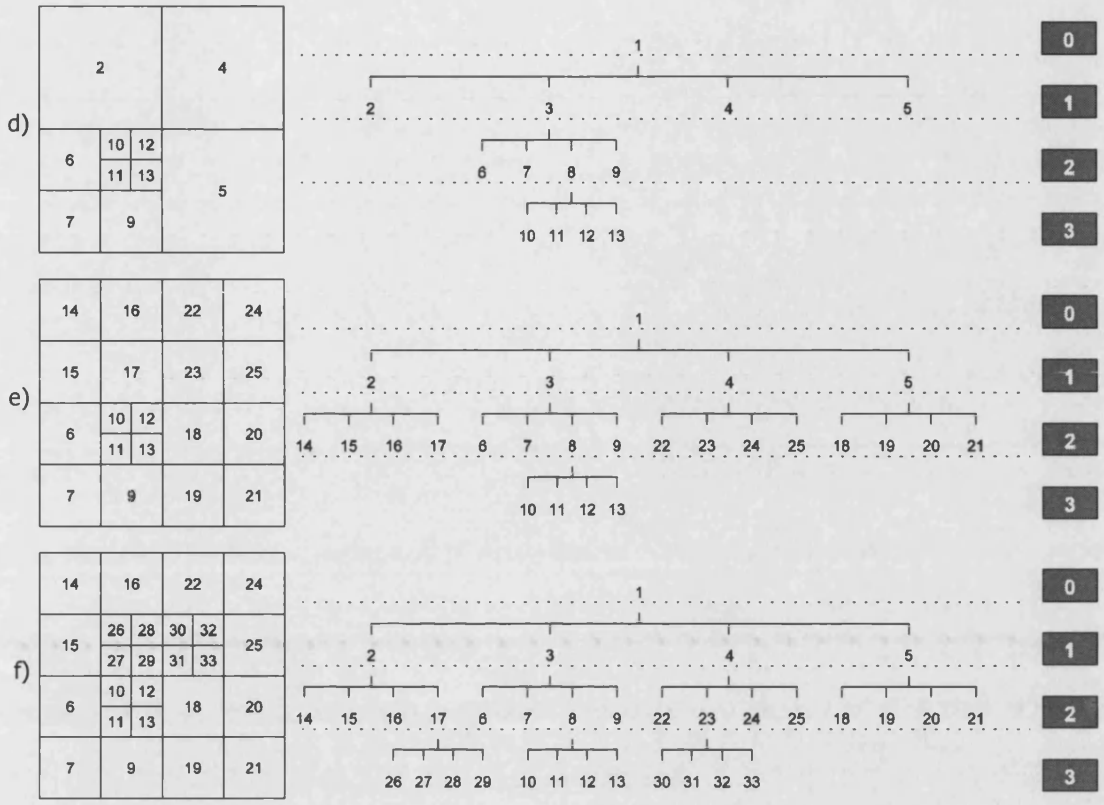


Figure 3.1. Steps in the generation of a quadtree grid – grid, tree and division levels. Seeding points A-D are denoted by •

of cells 2-5. As cell 3 spawns four offspring in the form of cells 6-9 , another branch is added to the tree in Figure 3.1c. As cells 6-9 are the children of one of its children, cell 1 is said to be the *grandparent* of these cells.

Figure 3.1 also displays the division level of the cells, that is, the number of divisions required to generate it. Genealogically, this is equivalent to the number of generations separating each cell from the primogenitor of the grid, cell 1. Although information regarding all cells in the grid is kept during the execution of the code in order to allow fast neighbour finding and grid manipulation techniques, only childless cells (called *leaf* cells) store values of the independent variables. For the grid in Figure 3.1f leaf cells comprise a total of 25 cells, i.e. cells 6, 7, 9-16, 18-22, and 24-33.

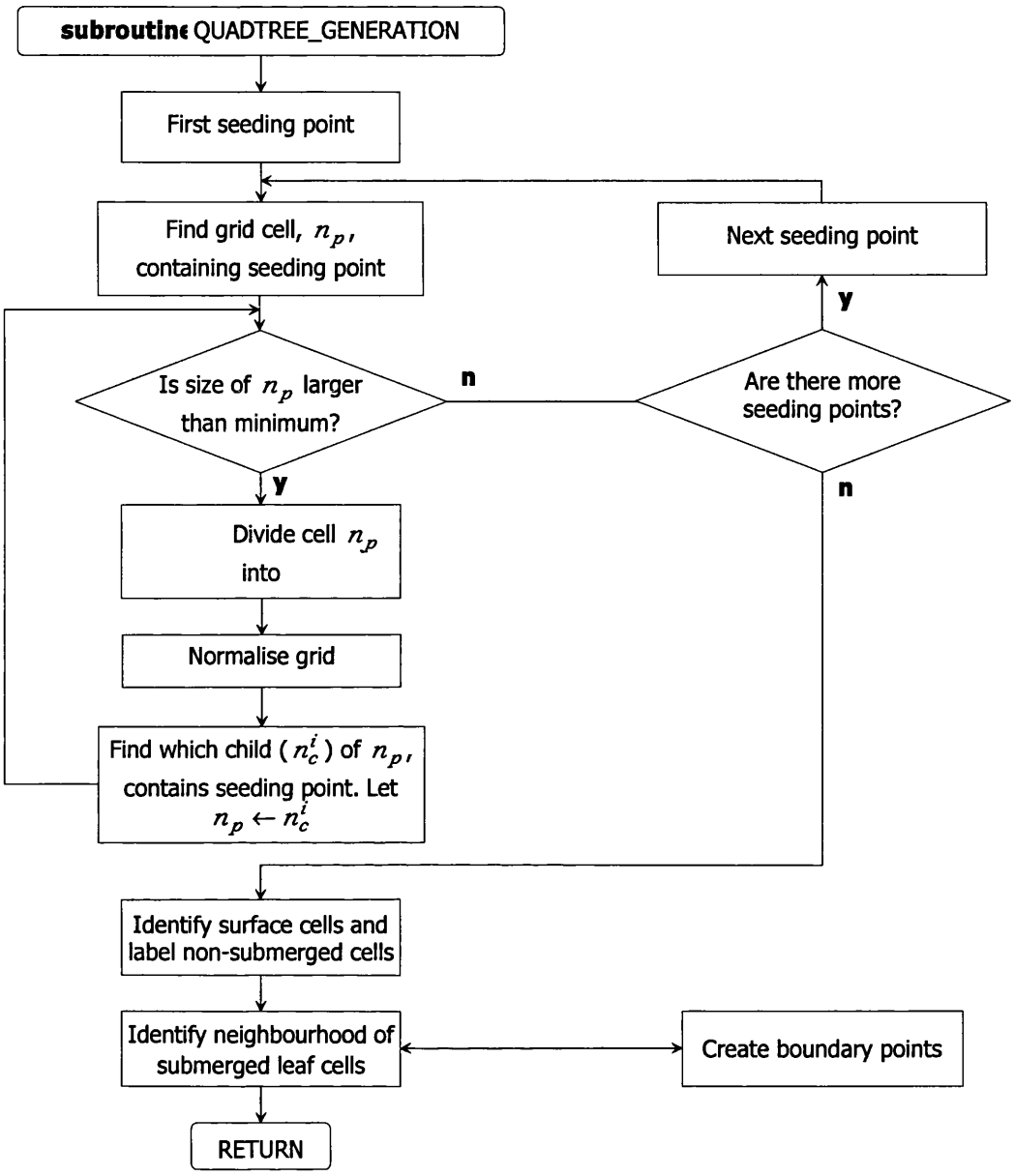


Figure 3.2. Algorithm of quadtree grid generation

The generation of quadtree grids is controlled by two sets of data: a) the minimum, l_b, and maximum, l_m, division levels (this is equivalent to specifying the maximum and minimum cell size), and b) the positions of a set of points, called *seeding points* (labelled A to D in Figure 3.1a) which inform the grid generation algorithm of the domain regions where higher resolution is desired. Whilst the former controls the overall quality of the grid, the latter details its specific topology. The algorithm demonstrated in Figure 3.1

considers each seeding point in turn and identifies the grid cell containing it (the code for the identification of the cell is shown in Figure B.7 in Appendix B). The cell will then be divided if its size is larger than the minimum size desired. Its children are in turn searched to locate the one containing the seeding point in question and more divisions are performed until the maximum division level is reached. Once all seeding points have been considered, the algorithm searches the grid for any leaf cells with a division level lower than the specified base level, l_b,and recursively divide these until l_b is reached.

A flowchart of this grid generation algorithm is shown in Figure 3.2.

A drawback to this method is that the maximum level of refinement is attained wherever a seeding point is placed. To overcome this, the division criterion may instead be based on seeding point density. For example, a cell may be selected for division if it contains more than a given number of seeding points, usually 1. Maximum and minimum cell sizes may be used as additional criteria, if desired. Figures 3.3a and 3.3b show the grids generated

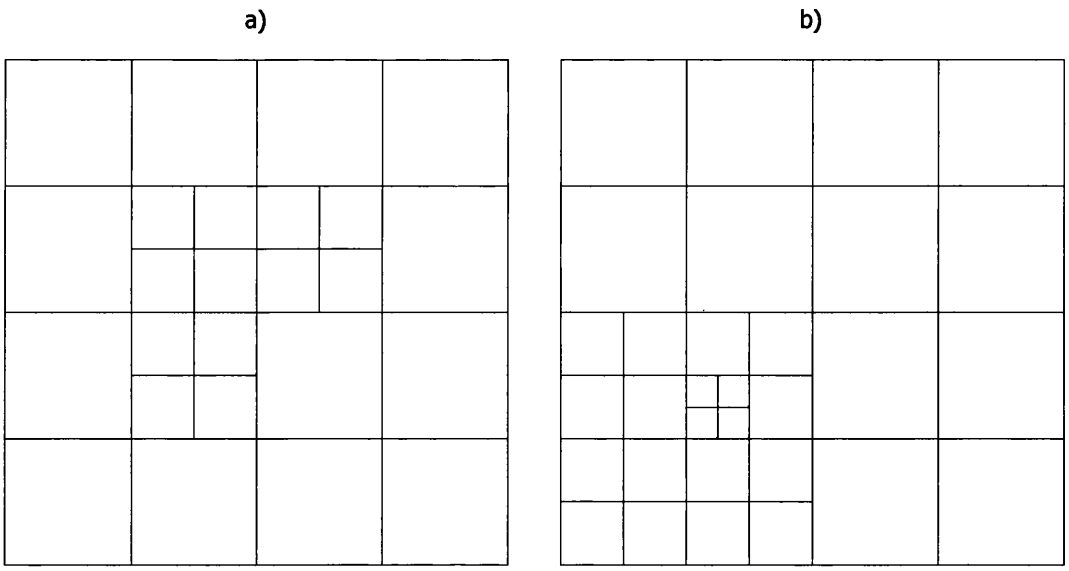


Figure 3.3. Effect of cell division criteria for the same set of seeding points:
a) Maximum division level criterion; b) Seeding point density criterion

by these two methods for the same set of seeding points A-D. For the seeding point density method, after the initial division of the root cell into cells 2-5, the proximity of seeding points A and B causes further divisions in the bottom left hand quadrant of the grid. As a result, this method requires a further division of cell 11 (cf. Figure 3.1e).

Additionally, it is desirable to control the maximum difference in size between neighbouring cells, a process called *grid normalisation*. This is beneficial for two reasons: a) it restricts the possible neighbour arrangements of a cell to a manageable number; and b) it reduces the discretisation errors that increase when distances between a cell and its neighbours vary greatly in the same arrangement. In this work, this limit was set at one division level, i.e. a cell may only have neighbours of the same division level, of one division level lower, or one division level higher. These conditions naturally require additional divisions to be performed in the grid, and can either be enforced in a subroutine tagged to the end of the grid generation algorithm or, alternatively, applied each time a cell division is made, as outlined in Figure 3.2. Enforcement of this restriction causes additional divisions on the grid of Figure 3.1d to be performed, resulting in the grid of Figure 3.1e. A full detailed description of the method used in the normalisation of a quadtree grid will follow once the cell numbering system is introduced.

An important component of the quadtree grid generation algorithm is the identification of the neighbours of a cell, also included in the algorithm of Figure 3.2. It is intuitive to use the tree structure of the quadtree grids to perform this task efficiently. For this purpose, it is necessary to use a cell numbering system that encapsulates the tree information. A description of

this system must therefore precede an account of the neighbour finding techniques.

3.2. Numbering System

To be able to quickly identify the size, ancestry, descendency and neighbourhood of a cell a reference numbering system must be used.

Again, a parallel with genealogy exists. In genealogical studies dating back to the 17th century, a numbering system was devised by the Spanish monk Jerome de Sosa (1676) to identify an individual's ancestors. This was later revised by von Strodonitz (1904) who popularised it under the name of Ahnentafel numbers. In these systems, the father was assigned a number equal to the double of that of the child in question, whilst the mother had the father's number with 1 added to it. Therefore, if the individual in question had reference number 1, his father's would be $2\times1=2$ and his mother's $2\times1+1=3$. It follows that his paternal grandmother would be identified by $2\times2+1=5$. The system is useful to identify lineage but is incapable of dealing with non-ancestral types of kinship since it can only move up the family tree, never downwards.

By contrast, the numbering system used in this work must allow for movement up and down the quadtree. It must also differentiate between siblings, in order to determine the position of a cell. In Chapter 2, the numbering systems of van Dommelen & Rundensteiner (1989) and Samet (1982) were discussed and the advantages of each method were remarked upon. Due to the fact that the latter is more economical in terms of storage than the binary system of van Dommelen & Rundensteiner, a version of the Samet system was devised for this work. Instead of employing equation (2.11)

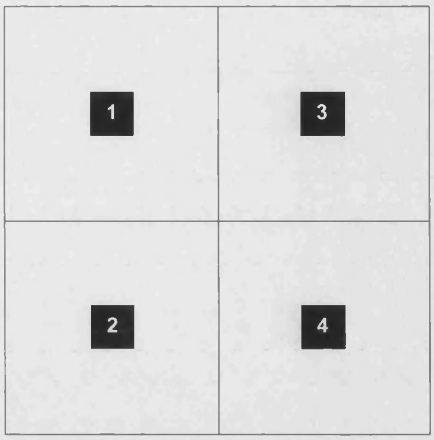


Figure 3.4. Quadrant positions

to generate the reference numbers of each cell, we propose here the following equation,

$$n_c^i = 4n_p + i, \tag{3.1}$$

where n is the cell reference number, subscripts p and c designate parent and children cells respectively and $i \in \{1,2,3,4\}$ is the quadrant position of each offspring according to the convention depicted in Figure 3.4. Equation (3.1) offers a few advantages over equation (2.11): namely, it produces smaller reference numbers which allow the creation of more refined quadtrees for the same integer length of n; and, more importantly, certain grid related operations, such as calculating n_p or i from n_c, require fewer arithmetic operations.

Figure 3.5 displays the cell reference numbers of the grid of Figure 3.1 at each stage of its generation. The difference between the cell numbering of Figures 3.1 and 3.5 must be emphasised. In the former, the numbers simply indicate the order in which the cells were generated (called *generation numbers*), whilst in the latter reference numbers are shown. For example, cell

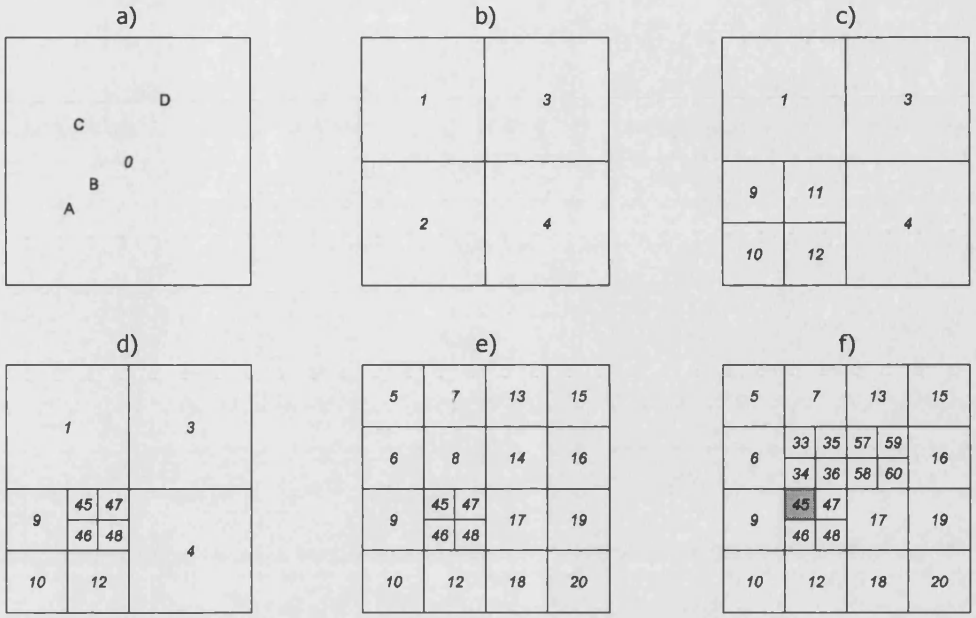


Figure 3.5. Cell reference numbers for the grids of Figure 3.1. Seeding points A-D are denoted by •

8 in Figure 3.1c has reference number 11 shown in Figure 3.5e. In turn, its children (cells 10-13) have reference numbers $n_c^i = 4\times 11 + i = \{45,46,47,48\}$, $i \in \{1,2,3,4\}$, as depicted.

Reference numbers allow the position, size, ancestry and neighbourhood of a cell to be determined using the procedures explained next.

3.2.1. Direct Ancestors

A parent's reference number can be obtained using the following expression,

$$n_p = \text{int}\left(\frac{n_c^i - 1}{4}\right), \tag{3.2}$$

where function 'int' designates the integer part of the real quotient on the right hand side, and n_c^i is the reference number of child i of n_p. By recurrently applying (3.2) to a cell's reference number all its direct ancestors'

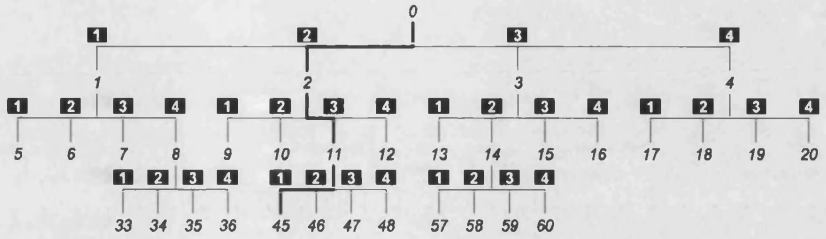


Figure 3.6. Tree path of cell 45. Reference numbers are shown in italic, white figures on black background denote quadrant positions.

reference numbers can be obtained, all the way up to the primogenitor cell. It can be observed that (3.2) is obtained by solving (3.1) for n_p with the 'int' function dealing with the fact that i is unknown, and the reference number needing to be offset by 1 to account for cells in quadrant position 4. As an example, the parent of cell 45 of Figure 3.5f is thus $n_p = \text{int}[44/4] = 11$, whilst its grandparent is $(n_p)_p = \text{int}[10/4] = 2$, and its first ancestor is $((n_p)_p)_p = \text{int}[1/4] = 0$, the root cell.

3.2.2. Division Level

It follows from Section 3.2.1 that the division level, l, of a cell can be determined by recurrently applying equation (3.2) to a cell's reference number until the number of the primogenitor cell is obtained, i.e. 0. The code for this functional procedure is shown in Figure B.1 in Appendix B. The number of times equation (3.2) needs to be applied to achieve this is equal to the cell's division level since it is equivalent to the number of times equation (3.1) was used in obtaining the cell's reference number.

Hence, the division level of cell 45 in Figure 3.5f is $L = 3$, as attested by Figure 3.6.

3.2.3. Quadrant Position

The quadrant position of a cell within its parent is obtained from

$$i_l = \mathrm{mod}\left(\frac{n_c^i - 1}{4}\right) + 1, \tag{3.3}$$

where i_l is the cell's quadrant position at division level l and function 'mod' represents the integer remainder of the division, defined as,

$$\mathrm{mod}\left(\frac{a}{b}\right) = a - b \cdot \mathrm{int}\left(\frac{a}{b}\right).$$

Equation (3.3) is again derived from (3.1). The addition and subtraction of 1 are necessary to deal with quadrant position 4, which would otherwise yield a zero value. If (3.3) is applied to the cell's direct ancestors the full tree path from the root to the cell in question can be determined.

For cell 45, at division level 3, (3.3) yields $i_3 = \mathrm{mod}[44/4] + 1 = 1$ and, using its parent's reference number, 11, $i_2 = \mathrm{mod}[10/4] + 1 = 3$. Finally, its grandparent, cell 2, has quadrant position 2. Hence, the cell's full tree path is i=[2 3 1]. This is attested by Figure 3.6. The code for this routine is shown in Figure B.2 in Appendix B.

3.2.4. Cell Size

The size of a cell is readily available once its division level has been determined. Figure B.3 in Appendix B displays the code for this calculation. Noting that the width (or height) of a cell is half of that of its parent it follows that a cell at division level 2 has a width, or edge length, equal to a quarter of that of the domain or root cell. If the domain is a square of width w_d, as will be the case throughout this work, a cell of division level l will have an edge length w equal to,

$$w = w_d / 2^l . \tag{3.4}$$

Cell 45 has an edge length $w = w_d/2^3 = w_d/8$, or one eighth of the width of the square domain.

3.2.5. Cell Co-Ordinates

To determine the co-ordinates of a cell, or, more accurately, the co-ordinates of its centre, its tree path must have been determined beforehand using the procedure described in Section 3.2.3. Depending on the quadrant position of a cell, its co-ordinates can be calculated with respect to the co-ordinates of its parent. For example, the x-coordinates of cells 1 and 2 in Figure 3.5b are less than that of the domain by an amount equal to half the edge length of these cells. It is therefore necessary to ascertain from a cell's quadrant position whether it is located to the left or right of its parent co-ordinates in the x-direction, or above or below them in the y-direction. The procedure can be interpreted as a transformation of the quadrant positions of cells into the binary numbering system used by van Dommelen & Rundensteiner (1989). In the x-direction this is performed using,

$$i_l^x = \mathrm{int}\left(\frac{i_l - 1}{2}\right), \tag{3.5}$$

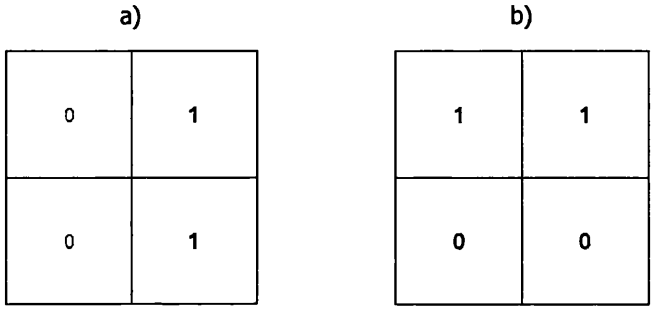


Figure 3.7. Transformed quadrant positions:
a) x-direction transformation; b) y-direction transformation

where i_l^x is the binary quadrant position in the x-direction as depicted in Figure 3.7a. The binary positions in the y-direction are obtained using,

$$i_l^y = \mathrm{mod}\left(\frac{i_l}{2}\right), \tag{3.6}$$

where i_l^y is the binary quadrant position in the y-direction as shown in Figure 3.7b.

Essentially, one is dividing the information contained in the integer array of possible quadrant positions, {1,2,3,4}, into two binary arrays {0,0,1,1} and {1,0,1,0}. If these transformations are applied to the full tree path of a cell, two distinct paths, one in each Cartesian dimension, are obtained. For example, for cell 45, the quadrant path i=[2 3 1], results in i^x=[0 1 0] and i^y=[0 1 1]. As previously mentioned in Chapter 2, the reference number of cell 45 in the binary system of van Dommelen & Rundensteiner (1989) is obtained precisely by interweaving the elements of these x- and y-arrays into a single one so that cell 45 would be referenced by n'_{45}=001101. In that system, however, +1 is added to each positional element of the array so that trailing zeros at the end of the array field are distinguished from x and y binary positions that have a zero value.

The co-ordinates of a cell of division level l with respect to its parent are then given by,

$$x_c = x_p + \frac{2i_l^x - 1}{2^{l+1}} = x_p + \frac{2\cdot\mathrm{int}\left(\frac{i_l - 1}{2}\right) - 1}{2^{l+1}}, \tag{3.7a}$$

$$y_c = y_p + \frac{2i_l^y - 1}{2^{l+1}} = y_p + \frac{2\cdot\mathrm{mod}\left(\frac{i_l}{2}\right) - 1}{2^{l+1}}, \tag{3.7b}$$

where subscripts p and c denote parent and child respectively. The co-ordinates of a cell are therefore determined by starting at the root cell whose co-ordinates are specified, and descending through the tree, successively applying equations (3.7). It should be noted that there is no need to store the binary paths of the cells as they are immediately available from equations (3.5) and (3.6).

As an example the co-ordinates of cell 45 of Figure 3.5 will be calculated, assuming the co-ordinates at the centre of the domain (i.e. the root cell 0) to be $x_d=y_d=0.5$. From the quadrant path of cell 45, $i=[2\ \ 3\ \ 1]$, equations (3.7) yield at the first division level, $i=1$,

$$x_2 = x_0 + \frac{2\cdot\mathrm{int}\left(\frac{i_1-1}{2}\right)-1}{2^{1+1}} = 0.5 - \frac{2\cdot\mathrm{int}\left(\frac{2-1}{2}\right)-1}{4} = 0.25,$$

$$y_2 = y_0 + \frac{2\cdot\mathrm{mod}\left(\frac{i_1}{2}\right)-1}{2^{1+1}} = 0.5 - \frac{2\cdot\mathrm{mod}\left(\frac{2}{2}\right)-1}{4} = 0.25,$$

which are the co-ordinates of cell 2, the grandparent of cell 45. At division level $i=2$, x_p and y_p in equation (3.7) take the value of the co-ordinates of cell 2 to give

$$x_{11} = x_2 + \frac{2\cdot\mathrm{int}\left(\frac{i_2-1}{2}\right)-1}{2^{2+1}} = 0.25 + \frac{2\cdot\mathrm{int}\left(\frac{3-1}{2}\right)-1}{8} = 0.375,$$

$$y_{11} = y_2 + \frac{2\cdot\mathrm{mod}\left(\frac{i_2}{2}\right)-1}{2^{2+1}} = 0.25 + \frac{2\cdot\mathrm{mod}\left(\frac{3}{2}\right)-1}{8} = 0.375,$$

the co-ordinates of cell 11, the parent of cell 45. Finally,

$$x_{45} = x_{11} + \frac{2\cdot\mathrm{int}\left(\frac{i_3-1}{2}\right)-1}{2^{3+1}} = 0.375 - \frac{2\cdot\mathrm{int}\left(\frac{1-1}{2}\right)-1}{16} = 0.3125,$$

$$y_{45} = y_{11} + \frac{2 \cdot \mathrm{mod}\left(\frac{i_3}{2}\right) - 1}{2^{3+1}} = 0.375 + \frac{2 \cdot \mathrm{mod}\left(\frac{1}{2}\right) - 1}{16} = 0.4375,$$

returns the co-ordinates of cell 45. The computer code for the procedures outlined above is displayed in Figures B.4 and B.5 in Appendix B.

3.3. Neighbour Finding

The hierarchical structure of quadtree grids and its associated reference numbering system allow the use of fast neighbour finding techniques based on simple arithmetic operations performed on the reference numbers of each cell. Because of the low computational cost of these operations, it is not necessary to store the numbers of the neighbours of each cell during computation, therefore allowing considerable savings in storage to be achieved, which are of particular significance when large highly refined grids are concerned. This section will describe the selected nomenclature of neighbours and the techniques devised to identify them using the adopted reference numbering system. Greaves (1995) describes analogous techniques for the numbering system of van Dommelen & Rundensteiner (1989). The problem of which neighbours to use in the discretisation of the governing equations will be dealt with in Chapter 4.

Excluding the case when a cell is adjacent to the edge of the quadtree grid, a cell can have two types of neighbours: face neighbours, which share a face with the cell in question; and corner neighbours which share a single vertex. If the cell is an interior cell, that is, if it is not adjacent to a domain boundary, it will have four or more face neighbours and between two and four corner neighbours. A general nomenclature for each of these can be developed based on the degree of kinship. Figure 3.8 displays the nomenclature chosen

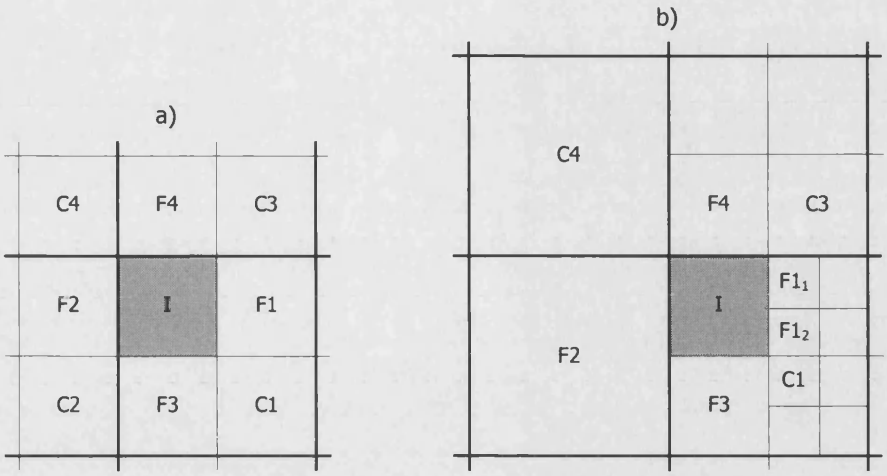


Figure 3.8. Nomenclature of neighbours of cell I for two different arrangements a) and b). Thick lines define parent cells, hence cell I has quadrant position 1 in both cases.

using cell I as an example in two different neighbour arrangements. The quadrant position of cell I determines the names attributed to each neighbouring panel. These can be divided into two categories, according to whether those neighbours are descendents of the parent of cell I or not. Those that are can be defined as,

- Face neighbour F1 is the sibling of I adjacent to it in the x-direction.
- Similarly, face neighbour F3 is the sibling of I adjacent to it in the y-direction.
- Finally, corner neighbour C1 is the sibling with which cell I share a single vertex.

It is worth mentioning that two F1 (or F3) neighbours may exist if these cells are divided, as illustrated in Figure 3.8b. However, there can only be one C1 corner neighbour, which is true for any corner neighbour. On the other hand, neighbouring cells that are not descendents of the parent of cell I, are defined by,

- Neighbour F2 is the other face neighbour in the x-direction (opposed to F1).

- Neighbour F4 is the other face neighbour in the y-direction (opposed to F3).
- Corner neighbour C2 is that located inside the larger cell adjacent to I in the x-direction.
- Corner neighbour C3 is that located inside the larger cell adjacent to I in the y-direction.
- Corner neighbour C4 is that located inside the larger cell with which cell I shares a single vertex (opposed to C1).

As with cells F1 and F3, face neighbours F2 and F4 may be divided. The techniques used to identify the reference numbers of all neighbours are detailed next using cell 45 of the grid in Figure 3.5f as an example. It should be noted that this cell is of division level $L=3$, and has tree path $i_{45}=[2 \quad 3 \quad 1]$.

NEIGHBOUR F1

To identify neighbour F1, it is necessary to travel transversely along the grid, more specifically laterally along the branch from which the cell in question stems. It is firstly necessary to identify the cell's parent since it will be nearest common ancestor of neighbours F1, F3 and C1. From (3.2), this is found to be cell 11, which has a tree path $i_p=[2 \quad 3]$, as expected. To select the sibling that faces the cell in the x-direction it is necessary to transform the quadrant position of the cell in question using,

$$i_{F1,L}=-1^{\text{int}[(i_{c,l}-1)/2]}\times 2+i_{c,l}. \tag{3.8}$$

Table 3.1 summarises the x-translation that equation (3.8) performs. For cell 45, which has $i_{45,l}=1$ the translation produces $i_{F1,l}=3$. It follows that the F1 neighbour of cell 45 is the child of cell 11 in quadrant position 3, i.e. the cell with reference number $n_{F1}=4\times 11+3=47$. If cell 47 had children then

Table 3.1 x-translation

$i_{c,l}$	$i_{F1,l}$
1	3
2	4
3	1
4	2

Table 3.2 y-translation

$i_{c,l}$	$i_{F3,l}$
1	2
2	1
3	4
4	3

Table 3.3 xy-translation

$i_{c,l}$	$i_{C1,l}$
1	4
2	3
3	2
4	1

its children in positions 1 and 2 would both be F1 neighbours of cell 45.

NEIGHBOUR F3

Similarly, neighbour F3 requires a translation of the quadrant position in the y-direction. This is accomplished using,

$$i_{F3,l} = i_{c,l} - (-1)^{\mathrm{mod}\left(\frac{i_{c,l}}{2}\right)}. \tag{3.9}$$

Table 3.2 displays the y-translations of the quadrant position. For cell 45, the translation produces $i_{F3,l}=2$. It follows that the F3 neighbour of cell 45 is the child of cell 11 in quadrant position 2, i.e. the cell with reference number $n_{F3}=11\times4+2=46$.

NEIGHBOUR C1

Finally, the sibling corner neighbour C1, requires both x- and y-translations in order to obtain its quadrant position. This is equivalent to applying (3.8) to the quadrant position of F3, or (3.9) to the quadrant position of F1. Table 3.3 summarises this combined translation. The C1 neighbour of cell 45 is therefore the fourth child of cell 11, i.e. cell 48.

NEIGHBOUR F2

A slightly more complex process is required to find the remaining neighbours due to the fact that the nearest common ancestor is unknown. To determine the reference numbers of these neighbours, it is necessary to ascertain at what stage in the tree path the cell position in the x-direction last changed. In other words, the highest division level at which the position of the cell's direct ancestors changes from the left (positions 1 or 2) to the right (3 or 4) or vice versa. This can be expressed mathematically by the highest division level, l_x, at which,

$$\text{int}\left(\frac{i_{c,l_x}}{2}\right) \neq \text{int}\left(\frac{i_{c,l_x-1}}{2}\right), \tag{3.10}$$

is true. Having determined this, x-translations are performed on the cell's tree path at all division levels between $l_x - 1$ and l.

Taking cell 45 as an example, with a tree path $i_c = [2 \;\; 3 \;\; 1]$, it can be seen that (3.10) holds true when $l_x = 3$. Applying (3.8) to levels 2 and 3 results in a tree path $i_{F2} = [2 \;\; 1 \;\; 3]$, which would result in a reference number of 39. However, as shown in Figure 3.5f, no such cell exists as cell 9 is childless. Therefore, the last division level of the tree path of F2 must be disregarded. It can then be seen that $i_{F2} = [2 \;\; 1]$ is the correct path for cell 9, as desired.

NEIGHBOUR F4

A similar process is required to obtain the reference number of neighbour F4. In this case, it is necessary to determine the highest division level in the tree path in which the quadrant position of the cell's ancestors changed from the top half (positions 1 or 3) to the bottom half (positions 2 or 4). This is achieved by finding division level, l_y, at which

$$\mathrm{rem}\left(\frac{i_{c,l_y}}{2}\right) \neq \mathrm{rem}\left(\frac{i_{c,l_y-1}}{2}\right). \tag{3.11}$$

The above condition states that the position of the cell's direct ancestors changes from the top (positions 1 or 3) to the bottom (2 or 4) or vice versa. The quadrant positions on the cell's tree path are then translated in the y-direction using (3.9) between division levels $l_y - 1$ and l. For cell 45, equation (3.10) is satisfied at $l_y = 2$. Hence, the tree path of neighbour F4 is found to be $i_{F4} = [1 \quad 4 \quad 2]$, the tree path of cell 34.

NEIGHBOUR C2

The tree path of neighbour C2 is obtained by performing x-translations between $l_x - 1$ and l, as for neighbour F2, with an additional y-translation at level l. Considering cell 45, the resulting path is therefore $i_{C2} = [2 \quad 1 \quad 4]$, which would be the child of cell 9 at quadrant position 4. As this cell does not exist in the grid, cell 45 does not have a C2 corner neighbour.

NEIGHBOUR C3

Conversely, neighbour C3 requires y-translations between $l_y - 1$ and l with an additional x-translation at level l. Hence, the path $i_{C3} = [1 \quad 4 \quad 4]$ points to the expected cell with reference number 36.

NEIGHBOUR C4

Finally, corner neighbour C4, which opposes the sibling C1, requires both x- translations between $l_x - 1$ and l, and y-translations between $l_y - 1$ and l. Neighbour C4 of cell 45 would have a path $i_{C4} = [1 \quad 2 \quad 4]$, in other words,

the child of cell 6 in quadrant position 4 as was to be expected. However, since cell 6 has not been divided, it itself is the C4 neighbour of cell 45.

BOUNDARY CELLS

Cells that lie adjacent to the edge of the quadtree can be identified by the fact that conditions (3.10) or (3.11) are not satisfied at any division level. If the cell lies adjacent to one of the vertical boundaries, condition (3.10) will be false throughout the tree path of the cell. If it is with a horizontal boundary that the cell shares an edge, then it will be (3.11) that always holds untrue. Both conditions will never be valid if the cell is located at the corner of the domain and, therefore, shares two faces with the edge of the quadtree.

3.4. Grid Normalisation

As was previously mentioned, grid normalisation is the process of enforcing a maximum size (i.e. division level) difference between neighbouring cells. This usually applies to face adjacent cells, but can also be extended, as is the case of this work, to corner neighbouring cells. From a programming point of view, this task may be performed after the grid has been divided about a set of seeding points or, alternatively, and arguably more elegantly, at each instance when a cell division occurs. A description of the method of grid normalisation will now follow using the grid shown in Figure3.5d and a maximum division level difference of 1 as an example. The reader is reminded of the grid generation process leading up to the aforementioned grid, described graphically in Figure 3.1a-d. Cells were created based on the location of the first seeding point. Up to and including division level 2, no grid normalisation is required. However, when cells of division level 3 are created, such as cells 45-48 in Figure3.5d, the grid normalisation criteria must be checked and

additional divisions may need to be performed. The algorithm cycles through these newly created cells, determines the reference numbers of potential neighbours two division levels larger in size (here the expression two generations removed (2GR) will be used) and divides these cells if necessary to reduce the difference in level to the prescribed 1. It should be noted that a cell will have at most three of these larger neighbours: two face neighbours, one in the x-direction and one in the y-direction; and one corner neighbour diagonally opposed (xy-direction). The process of obtaining the reference numbers of these cells will now be detailed.

x-DIRECTION 2GR NEIGHBOUR

The reference number of the x-direction 2GR neighbour of a cell is obtained by performing x-translations on the cell's quadrant path between division levels l_x-1 and $i-2$, provided $(l-2)\geq(l_x-1)$. If this condition is not satisfied, the cell does not have an x-direction 2GR neighbour. It can be seen that by stopping at level $i-2$, a cell two generations removed is obtained.

y-DIRECTION 2GR NEIGHBOUR

Similarly, the reference number of the y-direction 2GR neighbour of a cell is obtained by performing y-translations on the cell's quadrant path between division levels l_y-1 and $l-2$. Again, if $(l_y-1)>(l-2)$, the cell does not have a y-direction 2GR neighbour.

xy-DIRECTION 2GR NEIGHBOUR

The reference number of the xy-direction 2GR neighbour of a cell is obtained by combining both transformations, i.e. by performing x-translations on the cell's quadrant path between division levels l_x-1 and $i-2$ and

subsequent y-translations between division levels l_y-1 and $i-2$. If either $(l_x-1)>(l-2)$ or $(l_y-1)>(l-2)$, then the cell does not have an xy-direction 2GR neighbour.

Concerning the normalisation procedure carried out on the grid of Figure3.5d, of the four newly created children of cell 11, the algorithm first considers cell 45, that on quadrant position 1. As previously mentioned this cell has quadrant path $i_{45}=[2\ 3\ 1]$, $i=3$, $l_x=3$ and $l_y=2$. Since $(l_x-1)>(l-2)$, cell 45 has neither x- or xy-direction 2GR neighbours. On the other hand, a y-translation between $l_y-1=1$ and $i-2=1$ yields a quadrant path $i=[1]$ for the y-direction 2GR neighbour, i.e. cell 1. The algorithm descends through the quadtree along this path and verifies if cell 1 is divided. As this is not the case, cells 5-8 are created. The algorithm then proceeds to cell 46 which has $i_{46}=[2\ 3\ 2]$, $i=3$, $l_x=3$ and $l_y=3$. It follows that this cell has no neighbouring 2GR neighbours.

Cell 47 has quadrant path $i_{47}=[2\ 3\ 3]$, $i=3$, $l_x=2$ and $l_y=2$. The reference number of the x-direction 2GR neighbour is obtained by performing x-translations on division level 1. This yields a quadrant path $i=[4]$ which points to cell 4. As this cell has no children, it is divided and cells 17-20 are created. The reference number of the y-direction 2GR neighbour is obtained by performing y-translations on division level 1, which produces path $i=[1]$, i.e. cell 1. As this cell has already been divided, no action is taken. Finally, the quadrant path of the xy-direction 2GR neighbour is obtained by applying both x- and y-translations at division level 1. This yields $i=[3]$, pointing to cell 3, which is subsequently divided into cells 13-16.

Finally, cell 48 ($i_{48}=[2\ 3\ 4]$, $i=3$, $l_x=2$ and $l_y=3$) has no neighbouring y-direction 2GR neighbour. The reference number of the x-direction 2GR neighbour is found to be cell 4. As this has been divided already, no further divisions are necessary, and the grid normalisation at this stage is complete.

It should be mentioned that this process is carried out whenever cells of a division level higher than 2 are created. That is to say that, if during the process of normalisation cell divisions are made creating cells of division level 3 or higher, these cells are also put through the normalisation procedure, in a recurrent fashion.

The effect of grid normalisation is further illustrated by Figure 3.9, which shows two grids generated about the same set of seeding points, distributed to form a circle. The non-normalised grid of Figure 3.9a displays neighbouring cells of greatly different sizes. The highlighted grey cell has face neighbours with an edge length equal to an eighth of its own, whilst simultaneously having a larger neighbour with an edge eight times longer than its own. This implies that the discretisation expression for the grey cell in question would use cells separated by 6 division levels. This would significantly increase the discretisation errors of the solution and therefore is the main reason for implementing grid normalisation. By contrast, the discretisation expressions for the cells of the normalised grid of Figure 3.9b use cells separated by a maximum of 2 division levels (in cases where a cell has a neighbouring cell one division level higher and another one division level lower). Although normalisation implies the creation of additional cells and the consequent computational costs that this entails, the reduction in discretisation errors and increased stability of the numerical solver, offset this disadvantage.

Having discussed all the main elements of the grid generation algorithm, it is now possible to verify how the code generates the grid of Figure 3.5f. The seeding points used are shown in Figure 3.5a, labelled in the order they are considered. The maximum division level permitted is 3 and cells containing seeding points will be successively divided until this division level is reached, following the algorithm of Figure 3.2.

The first seeding point, A, is found to be located, naturally, in the root cell. As a result this cell is divided giving rise to Figure 3.5b. The algorithm now considers each of the newly created cells (1-4) sequentially until the one containing seeding point A is found, i.e. cell 2. As cell 2 has a division level smaller than 3 it is divided into cells 9-12. A search among these cells determines that cell 11 contains seeding point A, resulting in an additional cell division into cells 45-48. Since division level 3 has been reached grid normalisation must be enforced. The y-direction 2GR neighbour of cell 45 is found to be cell 1 and, as it has no children, it is divided. The same applies to the 2GR neighbours of cell 47, i.e. cells 4 and 3 in this order. The algorithm now returns to cells 45-48, and verifies which contains seeding point A. However, since the maximum division level has been reached no more cell divisions are performed for this seeding point. The next seeding point is then considered. Seeding point B is found to be located in cell 46. As this cell is of division level 3, no new cells are created. By contrast, seeding point C causes the generation of cells 33-36. These are put through the grid normalisation routine, but new divisions are not necessary. Finally, the last seeding point brings about the division of cell 14 into cells 57-60. Grid normalisation again yields no new cells. The cell generation numbers of Figure 3.1 confirm the order in which the grid cells were generated.

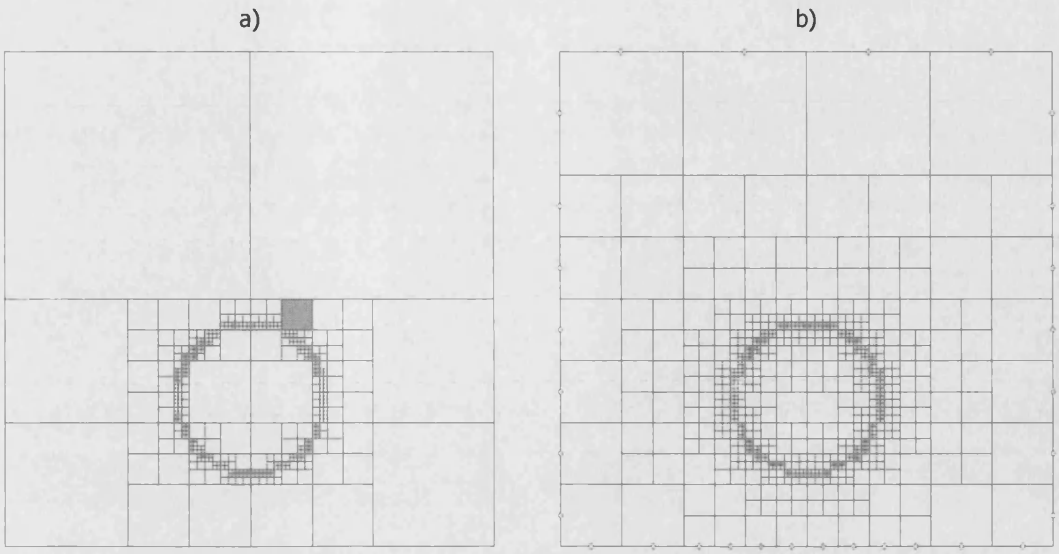


Figure 3.9. Comparison between a) non-normalised and b) normalised grids. Boundary points [◊] are also shown in figure b)

3.5. Free Surface Grid

The physical problems modelled in this work require fluid domains confined by two types of boundaries: the moving free surface boundary, and the rigid stationary boundaries such as the rigid walls of the numerical tank. During the grid generation stage these are addressed in different ways.

As the rigid walls coincide with the edge of the quadtree, the identification of the cells adjacent to these boundaries follows the procedure described at the end of Section 3.3. Once a cell has been found to lie adjacent to a rigid boundary, a point is created on the boundary midway along the edge of the cell in question. Through these points, the boundary conditions at rigid walls are implemented. Figure 3.9b shows the set of boundary points for the quadtree shown.

The free surface boundary has important features which demand a distinct treatment from the one used with solid boundaries: a) it is a moving boundary whose position must be known accurately at each point in time, and b) it does not correspond to an edge of the quadtree, which requires that an

additional procedure be carried out to identify those quadtree cells which lie adjacent to the free surface and those that fall outside the fluid domain and are therefore not used in the solution of the governing equations.

In the method presented herein a mixed Lagrangian-Eulerian approach is employed in which the free surface is modelled by an ordered series of Lagrangian particles called surface markers superimposed on the underlying Eulerian quadtree grid, as shown in Figure 3.10. The issue of selecting a Lagrangian or Eulerian approach for the modelling of the free surface has been extensively debated in Chapter 2. Here, the choice of a Lagrangian formulation was made due to the following advantages: a) it ensures that the free surface remains sharply defined during the solution process; and b) it permits accurate implementation of the free surface boundary conditions since its position and orientation are explicitly known. The main disadvantage of this approach is the difficulty in dealing with complex wave phenomena such as surface break-up and merging of interfaces, since the reordering of the surface markers required is not a trivial task. Miyata (1986) employed a similar method which was able to cope with wave breaking phenomena and related air entrapment, but his method suffered from limited accuracy due to the low-order velocity calculation at the free surface. Furthermore, Miyata's (1986) free surface velocity calculation required extrapolation to cells outside the computational domain. These problems are tackled here by employing a new velocity calculation procedure which requires no extrapolations beyond the boundaries of the fluid domain, of which more in Chapter 4. It follows that quadtree cells outside the fluid domain are not needed during the solution stage and that a procedure is required to identify them. Furthermore, the code must also identify the cells whose centres lie immediately beneath the

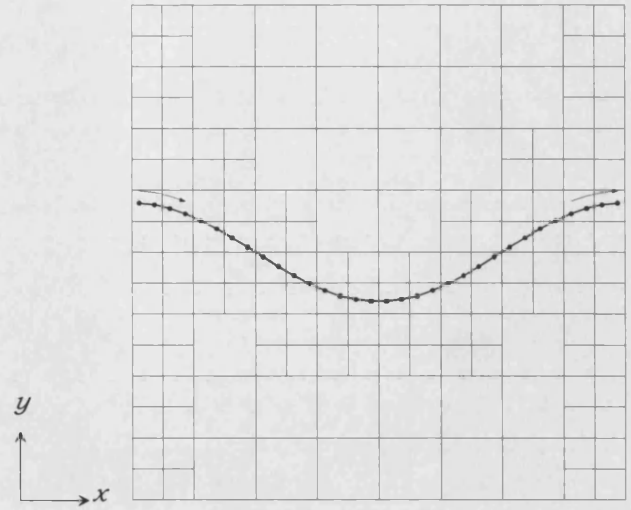


Figure 3.10 Surface markers [•] and orientation of free surface curve

free surface curve, as these are subject to a different discretisation procedure, as described in Chapter 4.

There are therefore three types of cells: 1) *non-submerged cells*, whose centres lie to the left as seen by an observer moving along the free surface and are not used during the solution of the governing equations; 2) *full cells*, which are submerged and entirely full of fluid; and 3) *surface cells*, which are also submerged but are adjacent to the free surface. It is through these cells that the free surface boundary conditions are applied.

To establish if a cell is part of the computational domain, the sequence of surface markers that defines the free surface must first be given an orientation. By convention, we chose it to be in the direction of increasing x, as illustrated in Figure 3.10. It is now possible to define that any cell centre that lies on the right 'side' of the curve lies inside the fluid domain, whilst those cells whose centre lie on the left of the curve are disregarded during the solution process. Accordingly, after the Eulerian quadtree has been generated the algorithm considers each surface marker in turn and locates the cell in which it is contained. For marker M_i in Figure 3.11a that is cell I. The centre

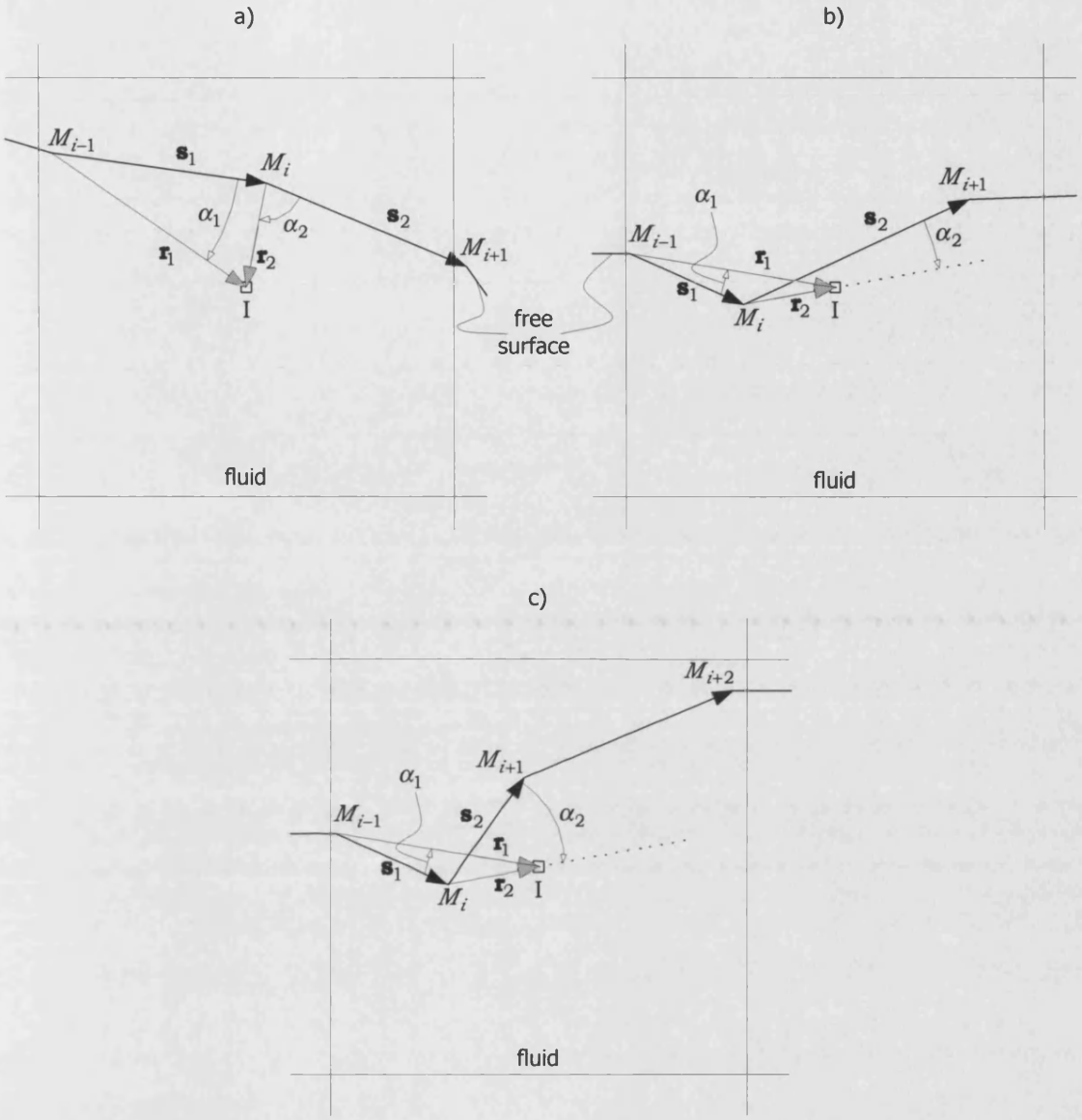


Figure 3.11. Assessment of submergence of the centre of cell I [ ] for three different surface geometries

of cell I is then judged to be below the surface curve if angles α_1 and α_2 are in the clockwise direction, or, equivalently, if the cross products between surface vectors $\mathbf{s}_1 = \overrightarrow{M_{i-1}M_i}$ and $\mathbf{s}_2 = \overrightarrow{M_iM_{i+1}}$ and the corresponding cell vectors $\mathbf{r}_1 = \overrightarrow{M_{i-1}I}$ and $\mathbf{r}_2 = \overrightarrow{M_iI}$ are negative, i.e. $\mathbf{s}_1 \times \mathbf{r}_1 < 0$ and $\mathbf{s}_2 \times \mathbf{r}_2 < 0$. This is a result of the choice of orientation of the surface curve shown in Figure 3.10. Furthermore, since cell I is immediately below the surface it requires a

different discretisation procedure than a deeply submerged cell would otherwise need.

There are, however, other surface configurations which require a careful choice of which cross product to use. This is illustrated in Figure 3.11b, where the cross products between the two nearest surface vectors $\mathbf{s}_1$ and $\mathbf{s}_2$ and the corresponding cell vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ are of opposite sign, as shown by the opposing direction of angles α_1 and α_2. In this case, the surface vector which crosses both co-ordinates of cell I is selected. Since $x_{M_i} < x_I < x_{M_{i+1}}$ and $y_{M_i} < y_I < y_{M_{i+1}}$, vector $\mathbf{s}_2$ is used and cell I is consequently labelled a surface cell as $\mathbf{s}_2 \times \mathbf{r}_2 < 0$.

An additional case may occur in which the two surface vectorsyield cross products of opposite sign, but neither crosses both co-ordinates of the cell centre. This is illustrated in Figure 3.11c. When this is the case the vector closest to the centre of the cell is used, i.e. vector $\mathbf{s}_2$ in Figure 3.11c.

These techniques essentially determine the surface vector, in the vicinity of cell I, for which a perpendicular straight line may be drawn, intersecting the vector and passing through the centre of I.

If, on the other hand, the centre of cell I is judged to be above the free surface curve, then its face neighbours are each put through the method just described to assess if they are surface cells. Since all face neighbours, and not simply the one immediately below, are checked for submergence, the algorithm can generate and correctly identify grid cells for any shape and orientation of the free surface, on the proviso that the surface markers are sufficiently close together.

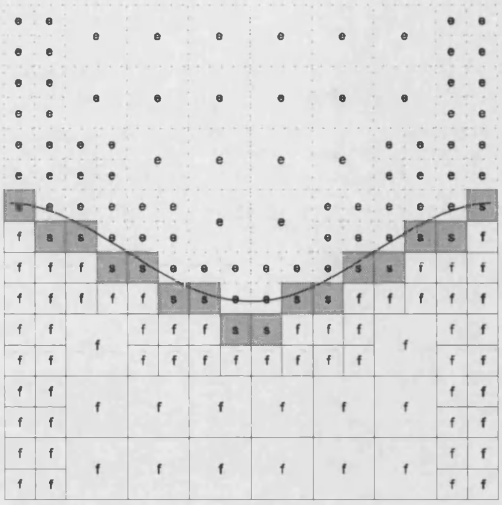


Figure 3.12. Labelled cells: full submerged cells, labelled 'f'; non-submerged cells, dotted and labelled 'e'; and surface cells, shaded and labelled 's'

As a result of this procedure, the surface cells of quadtree in Figure 3.10 are identified and labelled in Figure 3.12. They can be seen to form a contiguous chain of cells. This permits the construction of a fast recursive procedure in order to label non-submerged cells. It requires that a single non-submerged cell, I, be identified using a vector-based technique similar to that described above. This cell number is then used as input to a subroutine which labels I as non-submerged. The subroutine then cycles through the neighbours of cell I and recursively uses each unlabelled neighbour cell as an input to the same subroutine. Since the non-submerged portion of the quadtree grid is enclosed by a closed curve formed by the rigid boundaries and the contiguous string of surface cells, all non-submerged cells will be identified as such and the recursive subroutine can exit without the need of any additional programming constructs. The flowchart of this subroutine is displayed in Figure 3.13.

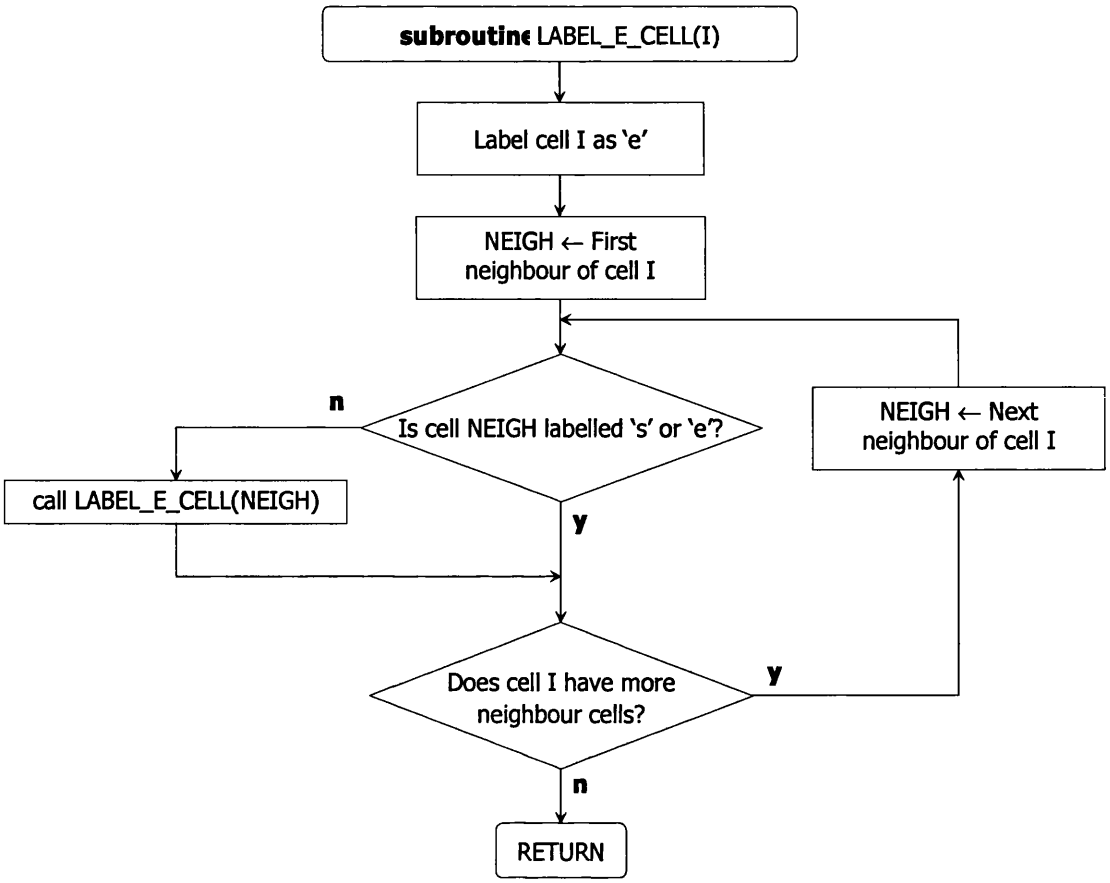


Figure 3.13 Flowchart of subroutine LABEL_E_CELL which labels non-submerged cells

To avoid markers becoming too spaced out in regions of high fluid velocity, their number is twice that of the surface cells. This was more accurate than introducing extra markers during the simulation, which, regardless of the interpolation method chosen, consistently introduced errors that lead to the simulation quickly halting.

3.6. Refinement

Higher grid refinement is desirable in the regions of the computational domain where independent variables vary greatly. In the current work, these are the areas where the gradient of the velocity potential is highest. Figure 3.14 displays the numerical solution of Laplace's equation as a three-dimensional surface at the start of the simulation of an irrotational water

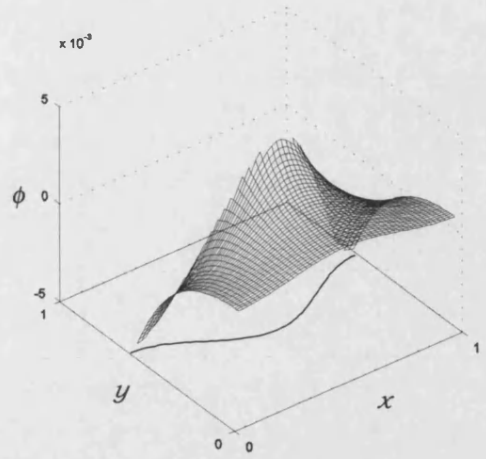


Figure 3.14. Typical shape of the velocity potential field, ϕ. The free surface curve has been projected onto the floor of the axes box.

wave. It can be observed that the gradient of the velocity potential is steepest near the free surface and along the vertical boundaries, whilst as the depth below the surface increases, the fluid velocities decrease. This phenomenon was observed by Longuet-Higgins (1953). With this in mind, it is desirable to use the surface markers that define the free surface curve as seeding points to the quadtree generation procedure. Additionally, seeding points are placed on the vertical walls of the numerical tank to provide additional refinement in this region.

Figure 3.15a shows a typical set of seeding points for a standing wave as used in this research. The resulting quadtree with a division level criterion $l_m = 8$ and $l_b = 5$ is also shown in Figure 3.15b. Non-submerged cells are not plotted as they are not part of the computational grid.

It could be suggested that additional grid resolution could be advantageous in the part of the domain $h-a \le y \le h+a;\ 0 \le x \le b$, since the gradient of the velocity potential is quite high in this area. However, Greaves *et al.* (1997) have shown that such an additional refinement does not visibly affect the solution when compared with a quadtree refined at the surface only

a) b)

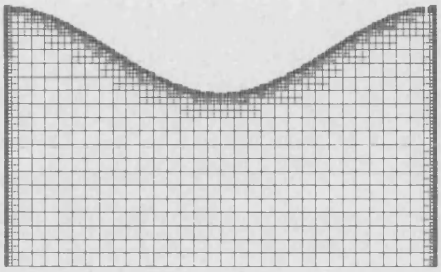


Figure 3.15.a) Set of seeding points and b) resulting quadtree for a standing wave. Maximum division level, $l_m = 8$, minimum division level, $l_b = 5$

as that of Figure 3.15. Refinement near the surface seems to have its most important effects in two ways: not only in the implementation of the Dirichlet condition for the solution of Laplace's equation for the velocity potential, but, more significantly, the velocity calculation at the free surface crucial to the kinematic boundary condition.

4. Solution Method

A solution method for unsteady flows may be split into two parts. The first consists of the solution of the governing equations in the spatial co-ordinates of the computational domain. The second part is the advancement of the solution in time.

To solve the governing equations, these must be replaced by a discretised model that approximates the exact solution as the grid is refined. There are three broad categories of method that achieve an algebraic representation of the governing differential equations in the whole fluid domain, namely, finite difference, finite volume and finite element methods. Higher-order techniques such as spectral methods are also important but are often developed with a specific class of problem in mind. The choice of method is dependent on two main factors: the type of equation to be solved and the type of grid used. In this work, a finite difference solver was chosen. The reasons for this choice along with the details of its implementation will be discussed in this chapter.

Prior to the advancement of the solution in time, the free surface velocities must be accurately calculated. This proved to be a crucial factor in the performance of the scheme. The discretisation in time has different characteristics, since it must be a marching scheme. For this reason, only finite difference schemes are usually used. Details of the time-stepping scheme chosen in this work are given at the end of this chapter.

4.1. Governing Equations for Irrotational Waves

Observations have shown that ocean waves travel long distances whilst retaining their shape and size. This indicates that the effects of viscosity are small. If, in addition, it is assumed that their motion was originally generated from rest, the principles of irrotational flow can be applied to accurately describe the motion of water waves and even more so that of standing water waves.

Furthermore, potential flow theory may be used in the description of free surface wave flow and its effect on marine structures provided the principal dimensions of the latter are comparable with the wavelength, in which case the forces on the structure are dominated by inertia effects and the viscous drag component may be neglected. Simulation of large amplitude wave motions requires a fully non-linear approach, in which the calculation grid deforms throughout the simulation to fit the moving free surface.

Considering the wave depicted in Figure 2.1, the following additional assumptions are made:

a. the fluid is incompressible

b. there is a rigid horizontal bed at $y=-h$

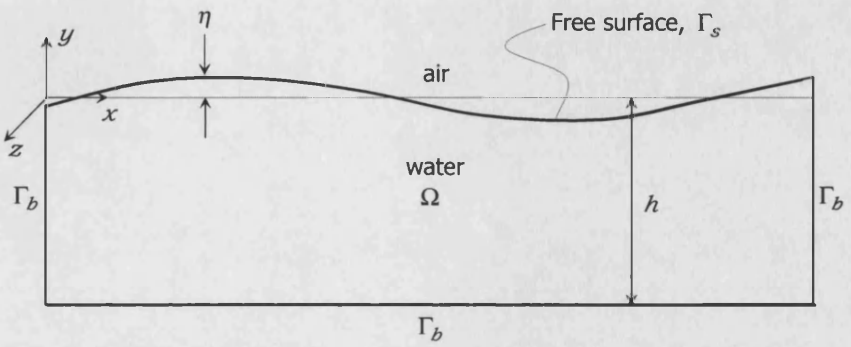


Figure 4.1 Water wave, computational domain and boundaries

c. boundary layer effects are negligible

d. shear stresses between water and atmosphere are negligible

e. the effect of air being set into motion by the motion of the wave is disregarded

For an incompressible fluid of constant density, the fundamental conservation laws that govern the flow in domain Ω_d are described by the Navier-Stokes equations:

Mass conservation: $$\nabla \cdot \mathbf{u} = 0, \tag{4.1}$$

Momentum conservation: $$\left(\frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla\right)\mathbf{u} = -\nabla\left(\frac{p}{\rho} + gy\right) + \nu\nabla^2\mathbf{u}, \tag{4.2}$$

where $\mathbf{u} = (u, v, w)$ is the velocity vector, $p(\mathbf{x}, t)$ is the pressure, ρ is the density, g is the gravitational acceleration, $\mathbf{x} = (x, y, z)$ is the position vector with y pointing vertically upwards, ν is the kinematic viscosity and t is time. Additionally, the vorticity vector $\mathbf{O}(\mathbf{x}, t)$ is defined as,

$$\mathbf{O} = \nabla \times \mathbf{u} \tag{4.3}$$

which may be shown to be equal to twice the local rate of rotation. Rotation of a fluid particle, however, can only be caused by a torque applied by shear forces on the sides of the particle. Since the fluid is assumed to be inviscid, such forces are absent and rotation cannot occur, and it follows that the vorticity vector,

$$\mathbf{O} \equiv 0. \tag{4.4}$$

For an inviscid irrotational flow, the velocity $\mathbf{u}$ can be expressed as the gradient of a scalar potential, Φ,

$$\mathbf{u} = \nabla\Phi. \tag{4.5}$$

It follows from equation (4.1), that continuity demands that

$$\nabla^2\Phi=0, \qquad \text{on } \Omega_d. \tag{4.6}$$

If the velocity potential is known, the pressure field can be determined from equation (4.2) using equation (4.4) and the fact that,

$$\mathbf{u}\cdot\nabla\mathbf{u}=\nabla\frac{\mathbf{u}^2}{2}-\mathbf{u}\times(\nabla\times\mathbf{u}),$$

to get,

$$\frac{\partial\mathbf{u}}{\partial t}+\nabla\frac{\mathbf{u}^2}{2}=\nabla\left[\frac{\partial\Phi}{\partial t}+\frac{1}{2}|\nabla\Phi|^2\right]=-\nabla\left(\frac{p}{\rho}+gy\right), \tag{4.7}$$

where the viscous term is ignored as the flow is considered inviscid.

BOUNDARY CONDITIONS

Two types of boundary conditions apply at the free surface: dynamic and kinematic. In what concerns the kinematic condition, one considers a hypothetical surface S anywhere in the fluid, moving with the fluid. If one follows each particle of surface S, the same particles will always form S and fluid inside the surface remains inside S. If $S(x,y,z,t)=0$, then as x, y, z, t vary for each particle, one remains on surface S, as

$$\frac{\mathrm{D}S}{\mathrm{D}t}=0, \tag{4.8}$$

for any surface S. $\mathrm{D}/\mathrm{D}t$ denotes the derivative following the motion of a particle. Taking S to coincide with the free surface between air and water, Γ_s where $y=\eta$, it may be defined by,

$$S=\eta(x,z,t)-y=0, \tag{4.9}$$

where η is a function describing the wave elevation. Substituting equation (4.9) in equation (4.8), yields

$$\frac{\partial \eta}{\partial t}+u\frac{\partial \eta}{\partial x}-v+w\frac{\partial \eta}{\partial z}=0, \qquad \text{on } y=\eta, \tag{4.10}$$

since η is independent of y and y is independent of all other variables. In terms of the velocity potential, Φ, equation (4.10) can be expressed as,

$$\frac{\partial \eta}{\partial t}+\frac{\partial \Phi}{\partial x}\frac{\partial \eta}{\partial x}-\frac{\partial \Phi}{\partial y}+\frac{\partial \Phi}{\partial z}\frac{\partial \eta}{\partial z}=0, \qquad \text{on } y=\eta, \tag{4.11}$$

which is the kinematic boundary condition at the free surface. If the analysis is restricted to two-dimensions on the xy-plane, equation (4.11) reduces to,

$$\frac{\partial \Phi}{\partial y}=\frac{\partial \eta}{\partial t}+\frac{\partial \Phi}{\partial x}\frac{\partial \eta}{\partial x}. \qquad \text{on } y=\eta, \tag{4.12}$$

If, on the other hand, S is part of a body surface such as the rigid horizontal bed in Figure 2.1 so that $S=-h(x,z,t)-y=0$, then equation (4.10) is instead,

$$\frac{\partial h}{\partial t}+u\frac{\partial h}{\partial x}+v+w\frac{\partial h}{\partial z}=0, \qquad \text{on } y=-h. \tag{4.13}$$

For a rigid level horizontal bed, equation (4.13) reduces to $v=0\equiv\partial\Phi/\partial y=0$, but for a moving object of arbitrary shape it generalises to,

$$\frac{\partial \Phi}{\partial n}=f, \qquad \text{on } \Gamma_b, \tag{4.14}$$

where f is the normal velocity of the body surface and n the direction normal to the body surface.

To derive the free surface dynamic boundary condition, equation (4.7) is integrated with respect to the space variables to yield,

$$-\frac{p}{\rho}=gy+\frac{\partial \Phi}{\partial t}+\frac{1}{2}|\nabla\Phi|^2+C(t), \qquad \text{on } y=\eta, \tag{4.15}$$

where $C(t)$ is some function of t which can be ignored by readjusting the velocity potential without affecting the velocity field. The first term on the right hand side of equation (4.15) is the hydrostatic pressure whilst the other two are dynamic contributions.

SURFACE TENSION

Once the velocity potential has been calculated by solving equation (4.6) in domain Ω_d, all terms on the right hand side of equation (4.15) can be computed. A question that remains open is the value of the pressure immediately underneath the free surface that appears on the left hand side. This value is dependent not only on the pressure acting on the free surface, which in this work is assumed to act with a constant magnitude along the surface curve, but also on additional fluid pressure required to balance the surface tension forces acting tangentially to the free surface.

The two-dimensional fluid element OABC shown in Figure 4.2, is bounded on ABC by the free surface segment ds. As $ds \to 0$, then $d\alpha = \sin d\alpha$ and the element can be regarded as an arc of a circle of radius r. Tangentially to the surface, there acts a tension whose magnitude is given by the surface tension coefficient, τ, which varies from fluid to fluid and is also dependent on temperature. It can be seen from Figure 4.2 that the surface tension acting on ds causes a force to act on the element along OB equal to $2\tau \sin d\alpha \approx 2\tau d\alpha$. This force has to be balanced by the net pressure on ds, so that

$$(p_a - p)ds = 2\tau d\alpha, \tag{4.16}$$

where p is the pressure immediately underneath the surface and p_a is the atmospheric pressure acting outside the surface. If the curvature of the surface is assumed to be small, then $d\alpha \to 0$ and, as a result, $2\tau d\alpha \to 0$. This implies that

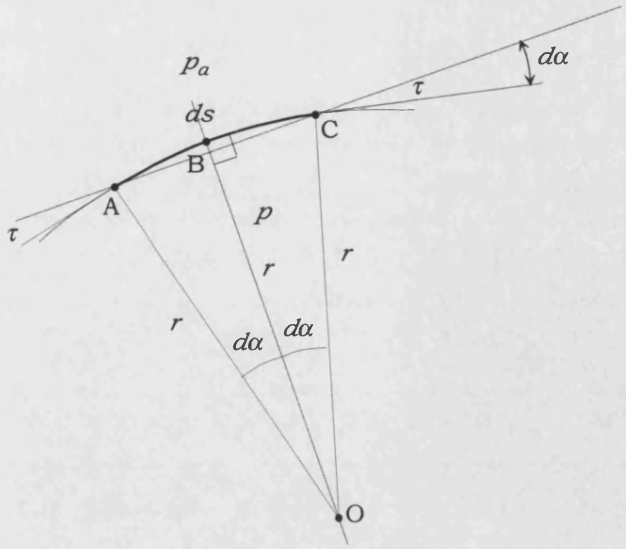


Figure 4.2 Balance of forces at the free surface

$$(p_a - p)ds = 0 \therefore p = p_a . \tag{4.17}$$

In other words, if surface tension is neglected the pressure immediately underneath the surface is equal to the atmospheric pressure. Variations in atmospheric pressure along the surface are disregarded and thus p is unvarying along the surface. The left hand side of equation (4.15) is therefore a constant, and the velocity potential can be adjusted to incorporate it without affecting the velocity field. Equation (4.15) thus reduces to,

$$g y + \frac{\partial \Phi}{\partial t} + \frac{1}{2}|\nabla \Phi|^2 = 0 , \qquad \text{on } y = \eta . \tag{4.18}$$

In two dimensions, the total derivative of the velocity potential is equal to,

$$\frac{D\Phi}{Dt} = \frac{\partial \Phi}{\partial t} + \frac{\partial \Phi}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial \Phi}{\partial y}\frac{\partial y}{\partial t} = \frac{\partial \Phi}{\partial t} + u^2 + v^2 . \tag{4.19}$$

Combining equations (4.18) and (4.19) produces the free surface dynamic boundary condition in Lagrangian form,

$$\frac{D\Phi}{Dt} = -g\eta + \frac{1}{2}\nabla\Phi \cdot \nabla\Phi , \qquad \text{on } y = \eta . \tag{4.20}$$

The non-linear problem can be solved according to the following sequence of steps. Two notations of the velocity potential are used: Φ is used to denote the exact solution of the governing equations, whilst ϕ, is employed to denote the solution to the discretised version of the governing equations:

a. The velocity potential at the free surface is assumed at the start of the simulation, $\Phi|_{t=0} = \varphi$. The initial shape of the surface is also assumed, $\eta|_{t=0} = \gamma$.

b. From the known values of ϕ at the free surface, and the Neumann boundary conditions given by equation (4.14), the velocity potential is calculated by solving equation (4.6) in the whole computational domain, Ω_d.

c. The velocities, u and v, are calculated at the free surface, so that the derivative of ϕ following the motion of a particle can be computed from equation (4.20).

d. The position and the velocity potential are updated using a time-stepping scheme to describe the problem at $t+\Delta t$.

e. The updated velocity potential and position of the free surface are then used in step b) and the process is repeated for the desired computational time.

In step a), at the start of simulations, the shape of the wave to be modelled is fed into step a). For example, the two-dimensional sinusoidal wave of Figure 4.3 is initially defined by the sinusoidal function,

$$y = \eta + h = a\cos\left(\frac{2\pi x}{\lambda}\right) + h, \qquad 0 \le x \le b, \tag{4.21}$$

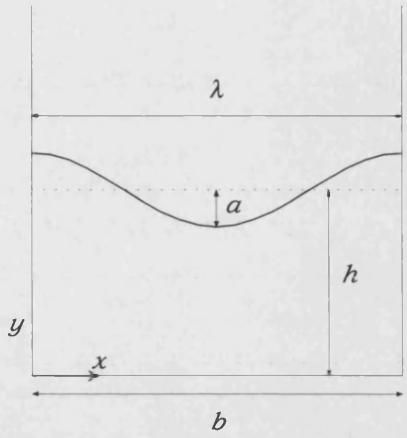


Figure 4.3 Parameters of a sinusoidal wave

where a is the amplitude of the wave, h is the mean water depth, λ is the wave length and b is the width of the computational domain, Ω_d. The origin of the axes has been placed on the bottom left hand corner of the tank, so that the co-ordinates of the surface Lagrangian grid and the quadtree Eulerian grid have the same origin. This fully non-linear approach can be used to model various standing waves in a tank. Examples of standing wave problems include the sloshing of liquids in a container and water on the deck of a ship. Motion can be started by specifying $\phi = 0$ at the free surface, which necessarily yields a zero velocity field, if all rigid boundaries are static. The initial geometry of the wave is prescribed so that γ is known. The wave is then initially accelerated by gravity alone, by the action of the linear term on the right hand side of equation (4.20). As the velocities increase, non-linear effects become significant due to the presence of the second term on the right hand side of equation (4.20).

The following sections will describe the implementation of the other steps in the scheme outlined above.

4.2. Finite Difference Discretisation

As mentioned at the start of this chapter, when the solution in the entire domain is sought, three broad categories of methods are available for the discretisation of the governing equations: finite element, finite volume and finite difference methods.

The use of finite element methods in quadtree grids involves a difficulty concerning the treatment of hanging nodes. Hanging nodes are vertices of smaller cells that are located along the face of a larger cell, as illustrated in Figure 4.4. Since variables are stored at the vertices of elements in finite element methods, a problem arises in the sharing of the nodal data amongst all three cells (elements). One strategy, used by Young *et al.* (1991), is to compute the value of the independent variables at these nodes as the average of the values at the extremities of the cell face, in order to ensure continuity of the element shape functions across the element faces. For the example shown, $\phi_H = (\phi_A + \phi_B)/2$. In this way, hanging nodes are not true degrees of freedom, and this averaging procedure introduces a residual error in the

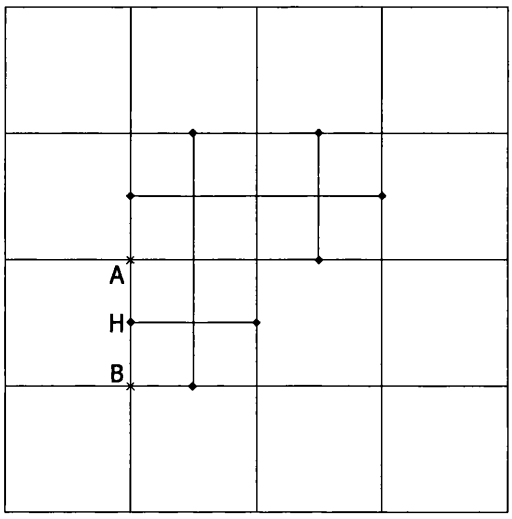


Figure 4.4 Hanging nodes, ♦

governing equations at ϕ_H which must be distributed to the connected nodes. A consequent degrading of the convergence rate is observed. An alternative strategy, employed for example by Jeong & Yang (1998), incorporates the averaging formula into the stiffness matrices of the elements. The disadvantage is that there is a loss of uniformity in these matrices and loss of vectorisation.

A radically different approach is to triangulate the grid, in such a way that all hanging nodes are eliminated. This approach has been employed, for example by Greaves *et al.* (1997) who constructed the triangular elements by joining the centres of three neighbouring square cells. They nevertheless found that additional triangulations, along with stretching and merging of triangular elements, was required to ensure boundary fitting and overall mesh quality. The main disadvantage is the loss in quadtree structure of the grid, rendering it less amenable to the implementation of the multigrid technique.

Finite volume methods have also been used with quadtree grids although mostly in the solution of strongly convective viscous flows. Examples include De Zeeuw & Powell (1993) and Greaves (1995). In two-dimensional finite volume methods, the integration of the flux gradient terms over the area of each control volume is replaced by line integrals around the edges of the volume. It is necessary to compute either the gradients at the midpoints of each of the edges of the volume, or the values of the variables at the midpoints from the gradient at the centre of the volume, depending on the formulation. When highly irregular distributions of neighbouring volumes occur, specially prevalent in the vicinity of an irregular boundary such as the surface Lagrangian mesh, the evaluation of these gradients may lead to less accurate approximations. De Zeeuw & Powell (1993) propose a scheme that maintains

accuracy provided at least three non-collinear neighbours are used in the evaluation of gradients. However, they found that this was prone to generating values that fell outside the bounds of the data used in the path integral itself. As a result, they felt it necessary to impose a limiter on the computation of these variables or gradients, but observed a serious degrading in the convergence rate of the solver as a consequence.

By contrast, finite difference methods are of much easier implementation on quadtree grids, although discretisation near irregular boundaries requires some attention. Hanging nodes do not pose a problem, as all calculations are performed at the centre of cells. The main disadvantage of finite difference schemes is that conservation of conserved variables is not rigorously enforced. However, this is of much higher significance for strongly convective flows than for the standing wave flows simulated in this work.

For these reasons, a finite difference formulation was considered more suitable for the problem and grid at hand.

4.2.1. General Method

A general method of discretising Laplace's equation (4.6) using finite differenc Fletcher (1991) amongst others. Gáspár & Simbierowicz (1992) briefly discuss other differencing schemes for quadtree grids apart from the one presented here. In the method of unknown coefficients, it is first assumed that the discretisation expression for node I has contributions from the potential at I and each of its n neighbouring nodes, in this case 1 to 5, so that,

$$\left[\frac{\partial^2\Phi}{\partial x^2}\right]_I + \left[\frac{\partial^2\Phi}{\partial y^2}\right]_I \approx a_I\phi_I + \sum_{i=1}^{n} a_i\phi_i = q_I = 0, \qquad (4.22)$$

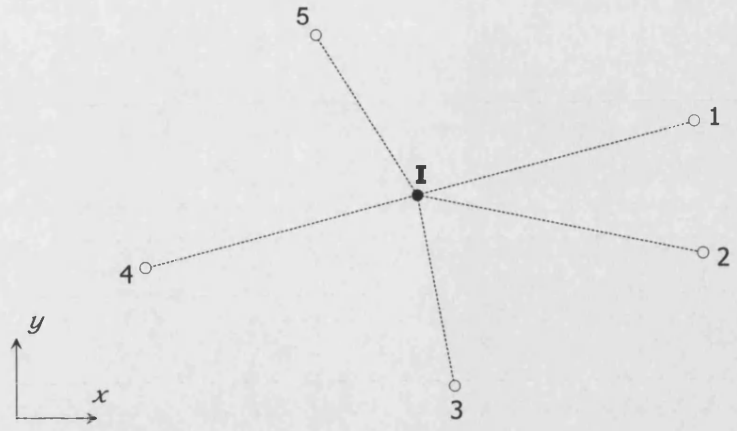


Figure 4.5. Random distribution of neighbouring nodes

where coefficients a_i and a_I represent the relative contributions of each node to the discretisation expression at node I. The source term q_I is equal to zero in case of Laplace's equation but it is convenient to introduce the source term as a variable at this stage. The notation Φ and ϕ exists to differentiate the exact solution to the partial differential equation and the solution to the discretised approximation, respectively. It follows from (4.22) that

$$\phi_I = \frac{q_I - \sum_{i=1}^{n} a_i \phi_i}{a_I}. \tag{4.23}$$

To determine the values of the coefficients that give the most accurate approximation to equation (4.22), the value of the velocity potential at each node is expressed by the Taylor series expansion,

$$\phi_i = \phi_I + \Delta x_i \left[\frac{\partial \phi}{\partial x}\right]_I + \Delta y_i \left[\frac{\partial \phi}{\partial y}\right]_I + \frac{(\Delta x_i)^2}{2!}\left[\frac{\partial^2 \phi}{\partial x^2}\right]_I + \frac{(\Delta y_i)^2}{2!}\left[\frac{\partial^2 \phi}{\partial y^2}\right]_I + \frac{\Delta x_i \Delta y_i}{2!}\left[\frac{\partial^2 \phi}{\partial x \partial y}\right]_I +$$
$$+ \frac{(\Delta x_i)^3}{3!}\left[\frac{\partial^3 \phi}{\partial x^3}\right]_I + \frac{(\Delta y_i)^3}{3!}\left[\frac{\partial^3 \phi}{\partial y^3}\right]_I + \frac{(\Delta x_i)^2 \Delta y_i}{3!}\left[\frac{\partial^3 \phi}{\partial x^2 \partial y}\right]_I + \frac{\Delta x_i (\Delta y_i)^2}{3!}\left[\frac{\partial^3 \phi}{\partial x \partial^2 y}\right]_I + \ldots, \tag{4.24}$$

where $\Delta x_i = x_i - x_I$, $\Delta y_i = y_i - y_I$ and $i \in \{1,2,\ldots,n\}$.

Replacing the value of the velocity potential at each neighbour in equation (4.22) by the above expansion yields,

$$\left[\frac{\partial^2\phi}{\partial x^2}\right]_I+\left[\frac{\partial^2\phi}{\partial y^2}\right]_I \approx a_I\phi_I+\sum_{i=1}^{n}\left(a_i\phi_I+a_i\Delta x_i\cdot\left[\frac{\partial\phi}{\partial x}\right]_I+a_i\Delta y_i\cdot\left[\frac{\partial\phi}{\partial y}\right]_I+\frac{1}{2}a_i(\Delta x_i)^2\cdot\left[\frac{\partial^2\phi}{\partial x^2}\right]_I+\right.$$
$$+\frac{1}{2}a_i(\Delta y_i)^2\cdot\left[\frac{\partial^2\phi}{\partial y^2}\right]_I+\frac{1}{2}a_i\Delta x_i\Delta y_i\cdot\left[\frac{\partial^2\phi}{\partial x\partial y}\right]_I+\frac{1}{6}a_i(\Delta x_i)^3\cdot\left[\frac{\partial^3\phi}{\partial x^3}\right]_I+$$
$$+\frac{1}{6}a_i(\Delta y_i)^3\cdot\left[\frac{\partial^3\phi}{\partial y^3}\right]_I+\frac{1}{6}a_i(\Delta x_i)^2\Delta y_i\cdot\left[\frac{\partial^3\phi}{\partial^2 x\partial y}\right]_I+$$
$$\left.+\frac{1}{6}a_i\Delta x_i(\Delta y_i)^2\cdot\left[\frac{\partial^3\phi}{\partial x\partial^2 y}\right]_I+\ldots\right) \qquad (4.25)$$

Equating the derivative terms on each side of equation (4.25), yields the following system of algebraic equations for coefficients a_i,

$$\begin{aligned}
&\sum_{i=1}^{n}a_i\Delta x_i=0\\
&\sum_{i=1}^{n}a_i\Delta y_i=0\\
&\frac{1}{2}\sum_{i=1}^{n}a_i(\Delta x_i)^2=1\\
&\frac{1}{2}\sum_{i=1}^{n}a_i(\Delta y_i)^2=1\\
&\frac{1}{2}\sum_{i=1}^{n}a_i(\Delta x_i)(\Delta y_i)=0\\
&\vdots
\end{aligned} \qquad (4.26)$$

The following additional equation

$$a_I=-\sum_{i=1}^{n}a_i\,, \qquad (4.27)$$

gives the value of the principal coefficient of node I defining all the terms of the discretisation expression (4.22). For closure, the algebraic system (4.26) must be made up of as many equations as the neighbours used in equation (4.22). As this number increases, the coefficients of higher order terms in equation (4.25) must be added to the system. When this is the case, the accuracy of the approximation is increased but the stability of the numerical procedure deteriorates. The truncation error of most discretisation expressions will be

proportional to a fraction of the cell's size. However when symmetry exists, the approximations become centred and are thus second order accurate.

4.2.2. Interior Cells

Interior quadtree cells are all submerged cells that are not surface cells and do not have a face lying on the edge of the quadtree.

By applying the method detailed in Section 4.2.1 to each of these neighbour configurations a series of discretisation expressions is applied. As an example, the quadtree cell I of Figure 4.6 will be considered. The cell's neighbours are identified using the compass notation shown and values of ϕ are stored at the centre of quadtree cells. The algebraic system for cell I is,

$$
\begin{aligned}
&a_{WS}+a_{WN}+a_E+a_S+a_N+a_I=0\\
&-\frac{3}{4}a_{WS}\cdot\Delta x-\frac{3}{4}a_{WN}\cdot\Delta x+a_E\cdot\Delta x=0\\
&-\frac{1}{4}a_{WS}\cdot\Delta y+\frac{1}{4}a_{WS}\cdot\Delta y-a_S\cdot\Delta y+a_N\cdot\Delta y=0\\
&\frac{9}{32}a_{WS}\cdot(\Delta x)^2+\frac{9}{32}a_{WN}\cdot(\Delta x)^2+\frac{1}{2}a_E\cdot(\Delta x)^2=1\\
&\frac{1}{32}a_{WS}\cdot(\Delta y)^2+\frac{1}{32}a_{WN}\cdot(\Delta y)^2+\frac{1}{2}a_S\cdot(\Delta y)^2+\frac{1}{2}a_N\cdot(\Delta y)^2=1\\
&\frac{3}{32}a_{WS}\cdot\Delta x\cdot\Delta y-\frac{3}{32}a_{WN}\cdot\Delta x\cdot\Delta y=0
\end{aligned}
$$

Since $\Delta x=\Delta y$ for the quadtrees of this work, the solution of the algebraic

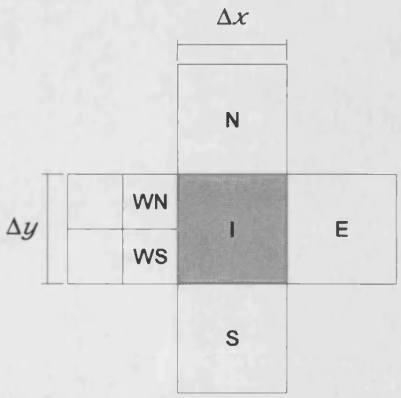


Figure 4.6. Example of neighbour arrangement of cell

system above is

$$a_{WS}=a_{WN}=\frac{16}{21(\Delta x)^2};\ a_E=\frac{24}{21(\Delta x)^2};\ a_S=a_N=\frac{20}{21(\Delta x)^2}\ \text{and}\ a_I=-\frac{96}{21(\Delta x)^2}.$$

Feeding these values into equation (4.23) gives

$$\phi_I=\frac{4\phi_{WS}+4\phi_{WN}+6\phi_E+5\phi_S+5\phi_N}{24}, \tag{4.28}$$

which is the discretisation expression for the neighbour arrangement of Figure 4.6.

Because quadtree grids are normalised, limiting the size difference between neighbouring cells, the number of possible neighbour arrangements an interior quadtree cell may have is limited to 12. These are compiled in Figures A.1-A.12 in Appendix A. For each arrangement, the corresponding discretisation expression is determined using the method just exemplified, and these are included in Appendix A.

It can be noted that several neighbour arrangements include more than five neighbours. It follows that more equations are required in algebraic system (4.26) to provide closure. In most arrangements however, symmetry can be used to reduce the number of unknowns. For example, the arrangement of Figure 4.7 is symmetrical about a central vertical line, which implies that $a_{WN}=a_{EN}$, $a_{WS}=a_{ES}$ and $a_{SW}=a_{SE}$. On the other hand, the arrangements in Figures 4.8 and 4.9 do not permit the use of symmetry. In these instances, additional equations are obtained from third order terms in equation (4.6) and added to the algebraic system. The discretisation expression is obtained using the set of coefficients that minimises these additional third order terms.

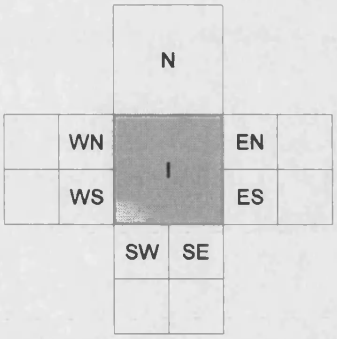


Figure 4.7 Discretisation arrangement 5

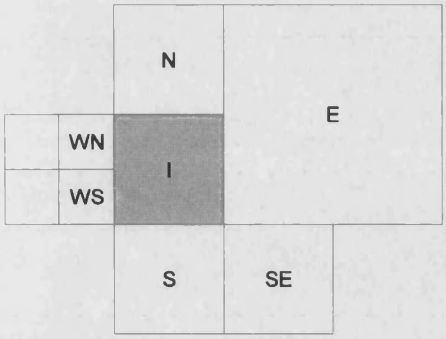


Figure 4.8 Discretisation arrangement 8

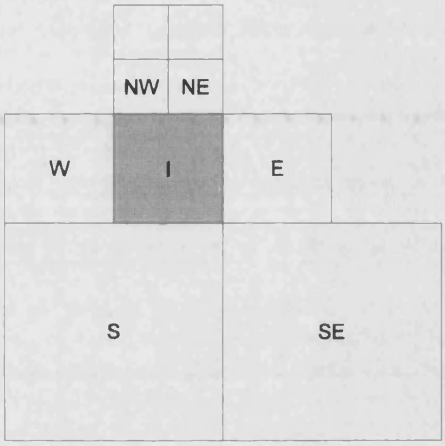


Figure 4.9 Discretisation arrangement 10

4.2.3. Boundary Cells

Cells that lie on the edge of the computational domain are called boundary cells. When these cells are generated during the grid generation procedure, the code creates additional boundary nodes located at the midpoint of the cell's face that lies on the edge of the computational domain, as discussed in Section 3.5. These are used to implement the boundary conditions prescribed at each boundary of the domain. Alternatively, mirror cells outside the domain could have been used.

It is apparent that new nodal neighbourhoods occur for boundary cells. Due to grid normalisation, these are limited to 11 extra arrangements, as

depicted in Figures A.13–A.23. The corresponding discretisation expressions are obtained in the same way as for interior cells, and are also included in Appendix A.

The boundary cell arrangements 16, 17, 18 and 23, shown in Figures 4.10-4.13 can be seen to include 6 or more neighbouring cells. Arrangements 17 and 18 are symmetrical about a horizontal line crossing points B and I allowing additional statements to be added to the algebraic system: for both arrangements $a_{NW} = a_{SW}$, $a_{NE} = a_{SE}$ with the additional identity $a_{EN} = a_{ES}$ for arrangement 18. Arrangement 23 is symmetrical about a diagonal line at 45°

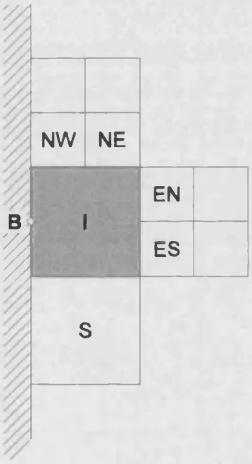


Figure 4.10 Neighbour arrangement 16

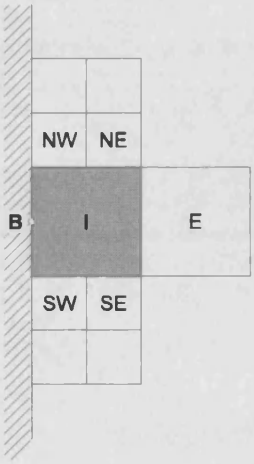


Figure 4.11 Neighbour arrangement 17

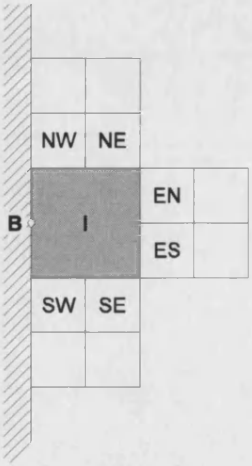


Figure 4.12 Neighbour arrangement 18

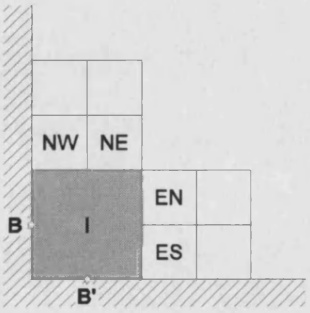


Figure 4.13 Neighbour arrangement 23

with the horizontal axis. This permits statements $a_{NW}=a_{ES}$, $a_{NE}=s_{EN}$ and $a_B=a_B$ to be used. Arrangement 16 in Figure 4.10, however, requires the use of third order terms in equation (4.25) to provide extra equations in the algebraic system. The discretisation expression thus determined provides the maximum average reduction of the third order derivatives in equation (4.25).

4.2.4. Surface Cells

Sections 4.2.2 and 4.2.3 describe the discretisation of nodal neighbourhoods that are bound to occur in a quadtree grid due to its structure and the criteria imposed during its generation. The discretisation expressions may therefore be computed beforehand and coded in to be applied to appropriate grid cells. However, there is a type of cell where the spacing between it and its neighbours does not conform with any predetermined distribution. This is the case of surface cells, located near the fluid's free surface, and defined by the fact that they require the use of surface markers in their discretisation expressions. The procedure for the identification of these cells was outlined in Section 3.5. Since surface markers are Lagrangian particles allowed to move freely, it is not possible to predetermine the discretisation of the governing equation at these cells.

Cell I, in Figure 4.14, will be used as an example of this process. The cell's neighbour arrangement is initially identified as arrangement 2, as it is surrounded by neighbours of the same size. However, since the centre of two of its neighbour cells, N and E, are above the free surface, no nodal values of ϕ are stored at these cells. As a result, surface markers must be used in their stead, when obtaining the discretisation expression for surface cell I. Since the positions of surface markers vary as the solution proceeds in time, the code must discretise Laplace's equation at each time step at each surface cell.

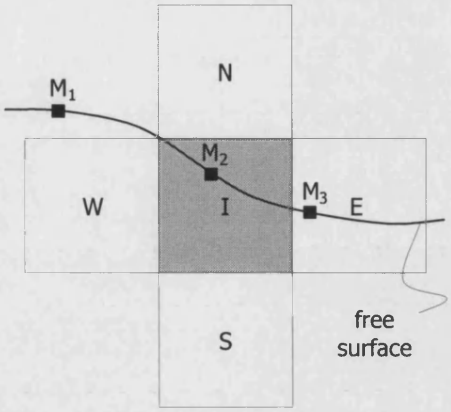


Figure 4.14. Neighbour Arrangement 24.
Black squares are surface markers.

This requires the solution of the algebraic system (4.26) which, for cell I will be, in matrix form,

$$\begin{bmatrix} \Delta x_W & \Delta x_S & \Delta x_{M_1} & \Delta x_{M_2} & \Delta x_{M_3} \\ \Delta y_W & \Delta y_S & \Delta y_{M_1} & \Delta y_{M_2} & \Delta y_{M_3} \\ \frac{1}{2}(\Delta x_W)^2 & \frac{1}{2}(\Delta x_S)^2 & \frac{1}{2}(\Delta x_{M_1})^2 & \frac{1}{2}(\Delta x_{M_2})^2 & \frac{1}{2}(\Delta x_{M_3})^2 \\ \frac{1}{2}(\Delta y_W)^2 & \frac{1}{2}(\Delta y_S)^2 & \frac{1}{2}(\Delta y_{M_1})^2 & \frac{1}{2}(\Delta y_{M_2})^2 & \frac{1}{2}(\Delta y_{M_3})^2 \\ \frac{1}{2}\Delta x_W \Delta y_W & \frac{1}{2}\Delta x_S \Delta y_S & \frac{1}{2}\Delta x_{M_1}\Delta y_{M_1} & \frac{1}{2}\Delta x_{M_2}\Delta y_{M_2} & \frac{1}{2}\Delta x_{M_3}\Delta y_{M_3} \end{bmatrix} \cdot \begin{bmatrix} a_W \\ a_S \\ a_{M_1} \\ a_{M_2} \\ a_{M_3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}. \tag{4.29}$$

The discretisation coefficients are computed by solving this system using Gaussian elimination with full pivoting, upon which the coefficient a_I can be found from equation (4.27). The value of the potential at the surface cell can then be estimated using

$$\phi_I = \frac{q_I - a_W\phi_W - a_S\phi_S - a_{M_1}\phi_{M_1} - a_{M_2}\phi_{M_2} - a_{M_3}\phi_{M_3}}{a_I}, \tag{4.30}$$

from which the value of ϕ_I can be updated. To ensure that inaccuracies do not arise in the discretisation of surface cells, cells that lie too close (at less than a tenth of the width of the surface cell) to surface markers

are discarded, even though they may be submerged. This is done to prevent distances in equation (4.29) becoming too small which would impair the quality of the Gaussian elimination solution of equation (4.29).

4.3. Iterative Scheme

Discretisation yields an algebraic system that approximates the original partial differential equation and can be expressed in matrix form as,

$$\mathbf{A}\phi - \mathbf{q} = \mathbf{0}, \tag{4.31}$$

where **A** is the coefficient matrix containing the discretisation coefficients a_i for all cells and **q** is the source vector equal to **0** in the case of Laplace's equation. As with most irregular grids, discretisation on quadtree grids creates a sparse and unstructured coefficient matrix, **A**. This is graphically shown in Figure 4.15 for a simple quadtree. It can be seen that not only is the coefficient matrix not diagonal, it is also not symmetrical. This means that a direct numerical solution is prohibitive, not only because it requires that the entire matrix be stored, but also because efficient methods such as those

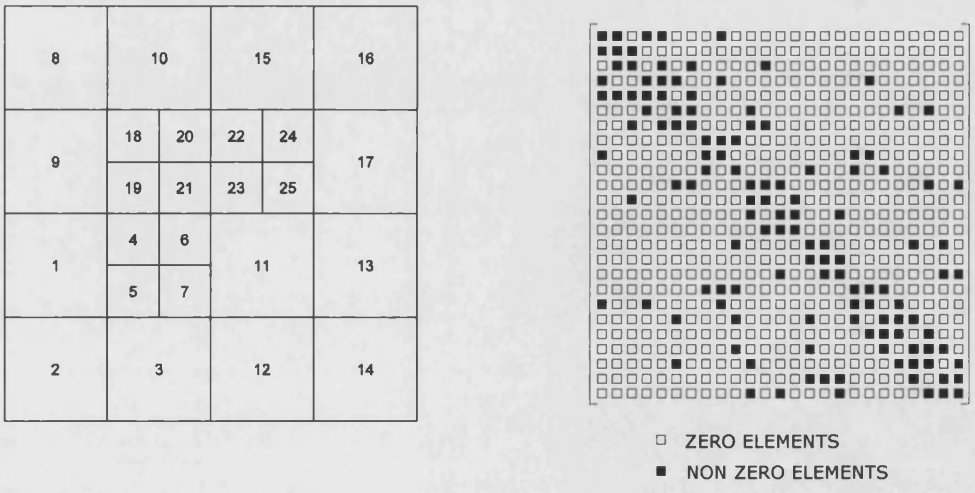


Figure 4.15. Quadtree and structure of corresponding coefficient matrix. Boundary points not included.

applied to tridiagonal matrices cannot be used. As a result, an iterative method is more appropriate. Due to the irregular nature of the grid, line by line methods are of awkward implementation and doubtful efficiency. Instead, the fully-explicit successive over-relaxation (SOR) point-by-point method is used.

In this method each computational cell is considered in turn according to a given order. Its neighbourhood is assessed according to the techniques discussed in Section 3.3. From this, the appropriate equation is selected from those in Appendix A. If the cell is a surface cell, the discretisation expression is determined as detailed in Section 4.3. In the SOR method, a relaxation factor is introduced to accelerate convergence by amplifying the correction to the potential at each iteration. Additionally, the values of the velocity potential at the neighbours of each cell used in the discretisation expression are the most recently updated values. Using the example of cell I in Figure 4.14, and assuming that the cells displayed are considered in the order W-I-S, equation (4.30) then becomes in the SOR method,

$$\phi_I^{n+1} = \omega_{SOR} \frac{q_I - a_W \phi_W^{n+1} - a_S \phi_S^n - a_{M_1} \phi_{M_1} - a_{M_2} \phi_{M_2} - a_{M_3} \phi_{M_3}}{a_I} + (1 - \omega_{SOR}) \phi_I^n , \qquad (4.32)$$

where the superscripts denote iteration number and ω_{SOR} is the relaxation factor. Although there are theoretical grounds to substantiate the choice of the optimum relaxation factor for regular grids (see Varga (1962), for example), it is difficult to extend this theory to irregular cases. As a result, values for ω_{SOR} have been chosen empirically in this work. These range between 1.1 and 1.25 with the higher values being used on larger grids.

The order in which the cells are considered during solution is worthy of some attention. To quickly impart the information on the boundary conditions

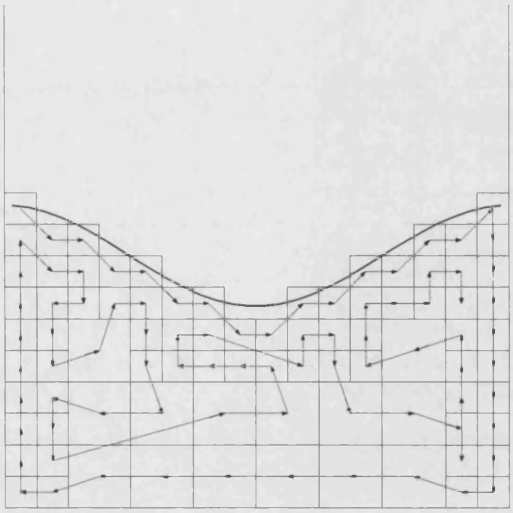


Figure 4.16 Order of solution of computational cells

to the whole of the domain, surface cells are considered first, followed by boundary cells in a clockwise perimeter around the fluid domain. A recursive algorithm is then used to order the interior cells, so as to maximise for each cell the number of terms in its discretisation expression with superscript $n+1$. The resulting order is shown in Figure 4.16. The ordering procedure is performed once at each time step after the generation of the new quadtree.

Finally, it is useful to alternate the cell solution order between the path shown and the inverse one so as to reduce the numerical bias that a sustained order tends to impart to the solution, which in symmetrical problems, for example, can cause asymmetry of the solution.

4.4. Free Surface Velocities

The computation of free surface velocities in methods for solving moving boundary problems has been addressed in very different ways according to the solution method employed. Accordingly, in boundary integral formulations, for example, it is possible to derive integral equations for the normal and tangential derivatives of the potential at each boundary (Longuet-Higgins &

Cokelet (1976), for example). The solution of these equations then permits the computation of the normal and tangential velocities at each element from the values of the potential at the other boundary elements.

By contrast, in field methods that discretise the entire domain using FD, FE or FV formulations to model viscous flows, it is common to extrapolate the pressure to empty cells above the surface in some way, and use these values to compute the surface velocities. Armenio (1997) gives one such example. Both these approaches are not available to full-domain methods for inviscid flows.

In the development of this work, the calculation of free surface velocities posed the most difficulty. This is partly due to the free unrestrained movement of the Lagrangian chain of particles, which define the surface, upon the fixed Eulerian grid. Preliminary attempts to model the motion of quasi-linear and non-linear waves showed that the accuracy of this method is strongly dependent on the method employed for calculating velocities at the free surface. This is particularly evident for the y-component v of the velocity, since, by definition, its calculation requires the use of submerged values of the potential which results in a one-sided approximation. The velocity component in the x-direction, u, presents less difficulty for waves of small or moderate amplitude since a centred approximation is usually possible using the values of ϕ at neighbouring surface markers. The several techniques attempted in the course of this work are described in the following two sections whilst a novel approach is presented in Section 4.4.3.

4.4.1. Calculation of v at Constant x

This technique, used for example by Turnbull (1999), requires that the surface markers be kept at known values of x throughout the solution. These

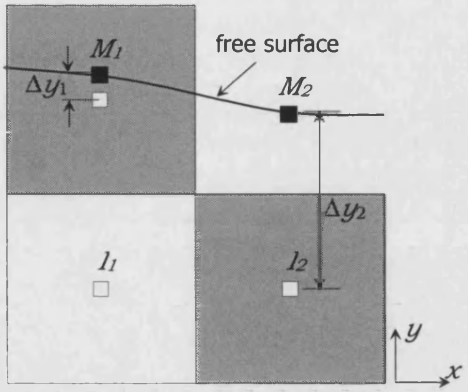


Figure 4.17. Velocity calculation at constant x. Surface markers, M_i, and grid nodes, I_i.

values of co-ordinate x are usually chosen to coincide with those of submerged grid nodes, so that surface markers are directly above these nodes. The restriction of constant x may either be implicitly incorporated in the Lagrangian formulation or, alternatively, after the surface marker positions and potential values have been updated at each time step, the markers can be returned to the original x values with new values of y and ϕ assigned using linear interpolation on the updated solution. Having thus constricted the horizontal movement of surface markers, arrangements similar to those of Figure 4.17 are produced in which surface marker, M_1, has the same x co-ordinate as node, I_1, and surface marker, M_2, has the same x co-ordinate as node, I_2. A first-order approximation to v from the surface marker and submerged node is then immediately available,

$$v = \frac{\partial \phi}{\partial y} \approx \frac{\phi_M - \phi_I}{y_M - y_I}. \tag{4.33}$$

The technique can be extended by imposing conditions on the grid generation algorithm to enforce that an additional grid node lies underneath nodes I_1 and I_2, permitting the use of a second-order accurate approximation.

4.4.2. Least Squares Approximation

An alternative method of calculating velocities at the surface, used by Greaves *et al.* (1997) and also discussed by Turnbull (1999), consists of using a least squares approximation to velocities u and v from the neighbouring nodal values of ϕ. From the definition of the velocity potential, the fluid velocity in any direction s can be expressed by,

$$\frac{\partial \phi}{\partial s} = \frac{\partial \phi}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial \phi}{\partial y}\frac{\partial y}{\partial s} = u\frac{\partial x}{\partial s} + v\frac{\partial y}{\partial s}. \tag{4.34}$$

In the least squares approach, u and v are estimated by calculating the pair of values that minimises the square of the distance to the exact solution of equation (4.34). Mathematically, the velocities are calculated by determining the values for which

$$q = \sum_{i=1}^{n}\left(u\left(\frac{\partial x}{\partial s}\right)_i + v\left(\frac{\partial y}{\partial s}\right)_i - \left(\frac{\partial \phi}{\partial s}\right)_i \right)^2 = \sum_{i=1}^{n}\left(ul_i^x + vl_i^y - \left(\frac{\partial \phi}{\partial s}\right)_i \right)^2 \tag{4.35}$$

is minimised and the summation is over the number of nodes used in the calculation. In equation (4.35), l^i is the position vector of neighbouring node i with respect to the surface marker, the x and y subscripts denote the horizontal and vertical components of l^i, and n is the total number of nodes used in the approximation. The approximation can include both surface markers and quadtree grid nodes. The minimisation procedure is performed by differentiating equation (4.35) with respect to u and v in turn and equating each to zero. This yields

$$\begin{aligned} \frac{\partial q}{\partial u} &= 2\sum_{i=1}^{n}\left[u\left(\frac{\partial x}{\partial s}\right)_i + v\left(\frac{\partial y}{\partial s}\right)_i - \left(\frac{\partial \phi}{\partial s}\right)_i \right]\left(\frac{\partial x}{\partial s}\right)_i = 0 \\ \frac{\partial q}{\partial v} &= 2\sum_{i=1}^{n}\left[u\left(\frac{\partial x}{\partial s}\right)_i + v\left(\frac{\partial y}{\partial s}\right)_i - \left(\frac{\partial \phi}{\partial s}\right)_i \right]\left(\frac{\partial y}{\partial s}\right)_i = 0 \end{aligned}. \tag{4.36}$$

4.4.3. Normal and Tangential Velocity Calculation

The techniques described above may be employed by the aforementioned authors because of the boundary fitted grids each used, which prevented sharp differences in the accuracy of the approximations for adjacent surface markers. In the present scheme, however, both suffer from severe inaccuracy under certain configurations of the free surface and underlying quadtree grid. This occurs when there is a marked difference in Δy for consecutive markers, as shown in Figure 4.17 where $\Delta y_1 << \Delta y_2$. This causes a discontinuity in the v-velocity field at the surface and a consequent step in the surface profile once the surface is advected. The error accumulates and soon leads to the calculation breaking down.

A solution to this problem is to calculate the velocities using an approximation with a constant distance throughout the surface profile. Here we propose a technique whereby this is achieved by first finding velocities normal and tangential to the surface. At a surface marker M_i the tangential

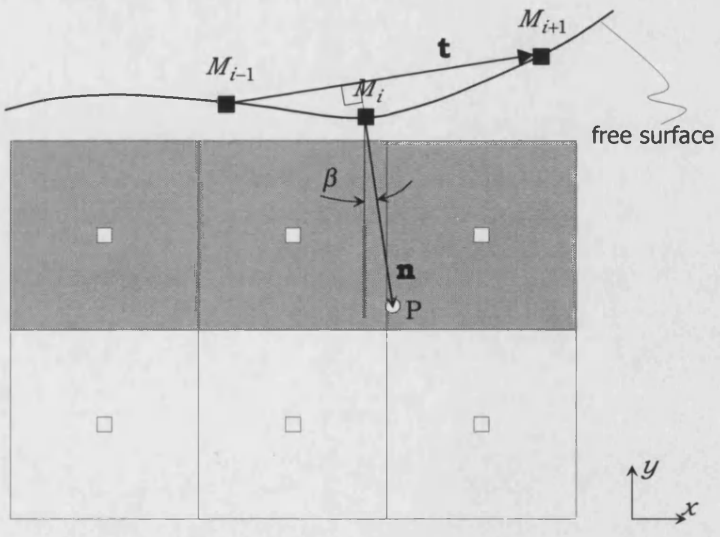


Figure 4.18. Calculation of normal and tangential velocities. Shaded cells are surface cells.

direction to the surface profile is estimated by using central differencing between markers M_{i-1} and M_{i+1}. A normal vector $\mathbf{n}$ is then used to locate a submerged point P, located at a distance from M_i equal to the edge length of the smallest cells in the grid, as shown in Figure 4.18.

The value of the velocity potential is calculated at P by discretising Laplace's equation using the method of unknown coefficients and the surrounding nodal values of ϕ. Because P always lies inside a polygon formed by known values of ϕ this calculation is much less susceptible to the distances between P and each of the surrounding nodes. The normal velocity, q_n, is then estimated using a first-order differencing expression,

$$q_n \approx \frac{\phi_P - \phi_{M_i}}{\|\mathbf{n}\|}, \tag{4.35}$$

positive in the direction of the fluid. The tangential velocity, q_t, is calculated along the free surface using the values of ϕ at the adjacent markers. This results in the expression,

$$q_t \approx \frac{\phi_{M_{i+1}} - \phi_{M_{i-1}}}{\|\mathbf{t}\|}, \tag{4.36}$$

where $\mathbf{t}$ is the vector connecting surface markers M_{i-1} and M_{i+1} used to estimate the tangent to the surface at M_i. The u- and v-velocities can then, if desired, be found from,

$$\begin{aligned} u &= -q_n \cos\beta + q_t \sin\beta \\ v &= -q_n \sin\beta + q_t \cos\beta \end{aligned}. \tag{4.37}$$

4.5. Time Stepping Scheme

Once the velocities at the free surface are known, the time derivative of the velocity potential is calculated from the dynamic boundary condition (4.20), and the solution can be advanced in time using a marching scheme. The simplest of these is the first-order Euler scheme:

$$\begin{aligned} \phi_{t+\Delta t} &= \phi_t + \Delta t \left.\frac{\mathrm{D}\phi}{\mathrm{D}t}\right|_t \\ x_{t+\Delta t} &= x_t + \Delta t \left.\frac{\mathrm{d}x}{\mathrm{d}t}\right|_t = x_t + u_t \Delta t . \\ y_{t+\Delta t} &= y_t + \Delta t \left.\frac{\mathrm{d}y}{\mathrm{d}t}\right|_t = y_t + v_t \Delta t \end{aligned} \tag{4.38}$$

However, it is often beneficial to employ a higher-order scheme, not just in terms of accuracy but also with respect to the stability of the scheme. Multi-step schemes which use the values of the independent variables at previous time-steps may be employed. This is the case of the Adams-Bashforth scheme described by,

$$\begin{aligned} \phi_{t+\Delta t} &= \phi_t + \left(\frac{3}{2} \left.\frac{\mathrm{D}\phi}{\mathrm{D}t}\right|_t - \frac{1}{2} \left.\frac{\mathrm{D}\phi}{\mathrm{D}t}\right|_{t-\Delta t} \right) \\ x_{t+\Delta t} &= x_t + \left(\frac{3}{2} u_t - \frac{1}{2} u_{t-\Delta t} \right) \\ y_{t+\Delta t} &= y_t + \left(\frac{3}{2} v_t - \frac{1}{2} v_{t-\Delta t} \right) \end{aligned} \tag{4.39}$$

4.6. Smoothing

When irrotational flows with a free surface are numerically modelled, the free surface has been observed to develop a high frequency seesaw instability of its profile. This has been found to occur regardless of the numerical technique employed to solve the governing equations. Boundary integral methods, such as those of Longuet-Higgins & Cokelet (1976) and Dold

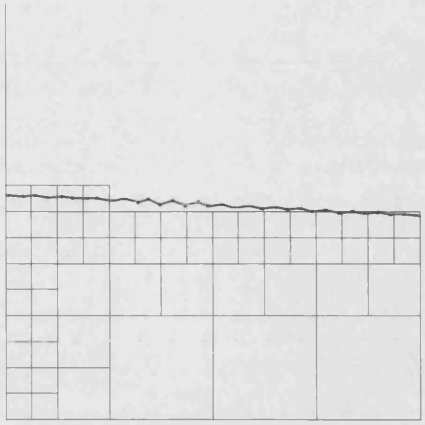


Figure 4.19 Detail of free surface profile, displaying seesaw instability.

(1992) have encountered this phenomenon, as well as the field methods of Wu & Eatock Taylor (1994) and Turnbull (1999). Some debate still exits over the cause of these instabilities, with some authors attributing it to numerical instabilities in the computation of free surface velocities, whilst some others consider them a result of lack of surface tension and fluid viscosity. If left unchecked, these instabilities will grow as the simulation progresses eventually causing the computation to collapse.

In the present method, the same seesaw profile of the free surface was found to develop during simulations, as displayed by the free surface detail displayed in Figure 4.19. To overcome this, the smoothing technique of Longuet-Higgins & Cokelet (1976) is employed. It approximates the surface profile by two Chebyshev polynomials: one representing the smooth part of the profile and the other being a quantity that oscillates with period $i = 2$ where i is the index of surface markers. A smooth function of the free surface variables is then obtained by disregarding the oscillatory polynomial. The coefficients of the polynomials are dependent on the number of surface markers used in the

smoothing process. In the present work, five markers were used, resulting in the smoothed function, $\bar{f}$,

$$\bar{f}_i = \frac{-f_{i-2} + 4f_{i-1} + 10f_i + 4f_{i+1} - f_{i+2}}{16}. \tag{4.40}$$

This function is applied at each surface marker $3 < i < n_i - 2$ to smooth surface elevation and velocity potential. This procedure has increased the robustness of the code and allowed longer simulations to be performed.

5. Multigrid Method

The unstructured nature of the coefficient matrices that result from using hierarchical Cartesian grids, restricts the choice of numerical solver used to solve the algebraic system. Since strong diagonal dominance is difficult to implement, quadtree solvers usually employ point-by-point solvers such as the Gauss-Seidel SOR solver described in Chapter 4.

The convergence rate of this type of iterative solver quickly deteriorates once the oscillatory errors have been smoothed after a few iterations. The multigrid method is a powerful technique to accelerate convergence, by bringing out the high-frequency error components on coarser grids.

This chapter will cover the details of multigrid implementation. It will start with an analysis of the convergence performance of the standard numerical method and the rationale behind the multigrid method. Two grid coarsening schemes are described along with the way the solution is represented on the coarse multigrid levels. The grid transfer procedures chosen are discussed.

5.1. Error Analysis of the Standard Solver

Before proceeding to analyse the convergence performance of the numerical solver, a few parameters must be defined first. The discretisation procedure yields an approximation to function Φ, and introduces a truncation error due to the neglect of higher order terms in the Taylor series approximations to the spatial derivatives of the velocity potential. This error shall be denoted by τ_d, and is defined as,

$$L(\Phi)=L_d(\Phi)+\tau_d \tag{5.1}$$

where L and L_d are symbolic operators representing Laplace's equation and the algebraic system after discretisation, respectively. The exact solution ϕ of the discretised equations must obey the following equation,

$$L_d(\phi)=\mathbf{A}\{\phi\}-\mathbf{q}=\mathbf{0}. \tag{5.2}$$

It differs from the exact solution of the partial differential equation by the discretisation error, ε_d, which is defined as,

$$\Phi=\phi+\varepsilon_d. \tag{5.3}$$

For iterative solvers, after n iterations, equation (5.2) will not be fully satisfied since only an approximate solution of ϕ has been achieved. Instead, there is a non-zero residual, $\mathbf{?}^n$:

$$\mathbf{A}\{\phi\}^n-\mathbf{q}+\mathbf{?}^n=\mathbf{0}. \tag{5.4}$$

At each iteration n there exists a convergence error, $\{\varepsilon_c\}^n$, defined as the difference between the iterated value of ϕ and the exact solution of the algebraic system,

$$\{\varepsilon_c\}^n = \{\phi\} - \{\phi\}^n . \tag{5.5}$$

From equations (5.5) and (5.3), the total error, *e*, of the numerical solution, i.e. the difference between it and the exact solution of the differential equation is equal to the sum of the discretisation and the convergence errors,

$$\{e\} = \{\Phi\} - \{\phi\}^n = \{\varepsilon_d\} + \{\varepsilon_c\}^n . \tag{5.6}$$

The iterative method is considered to have converged sufficiently once the convergence error has been reduced below some small value. However, the convergence error may not be computed as the exact solution to the algebraic system is obviously not available. Instead, another estimate of the convergence must be sought. Accordingly, subtraction of equation (5.4) from equation (5.2) gives the relation between the convergence error and the residual, called the residual equation:

$$\mathbf{A}\{\varepsilon_c\}^n = \mathbf{?}^n . \tag{5.7}$$

It shows that a reduction of the residual error is accompanied by a reduction

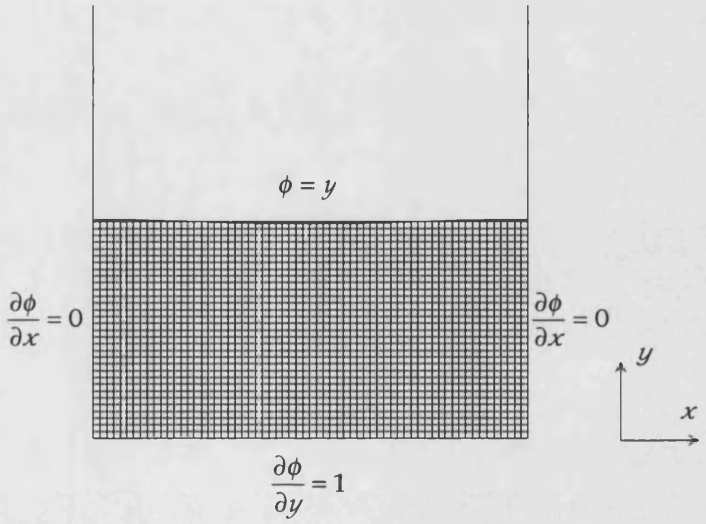


Figure 5.1 Regular quadtree grid (multigrid level *L*=4)

of the convergence error. Often, the residual is not used as a convergence measure since it requires the additional computational cost of solving equation (5.4) for ρ^n. However, since the multigrid method requires a solution of the residual equation (5.7), the computation of the residual field is necessary regardless. Consequently, a reduction in the residual norm is used as the convergence criterion. Specifically, the iterative solver is considered to have converged sufficiently once

$$\frac{\|\rho^n\|}{\|\rho^1\|} < 10^{-7} \tag{5.8}$$

is satisfied, unless otherwise stated.

It is of interest to analyse the behaviour of the error for the SOR solver. The case of Figure 5.1 will be used to illustrate this. The exact solution is $\phi = y$ and, accordingly, a Dirichlet condition is set on the surface whilst Neumann conditions are set on the rigid boundaries of the domain. A regular grid is used containing 2048 submerged cells. The initial guess is $\phi = 0$ at all internal and rigid boundary nodes. The numerical solution is displayed in

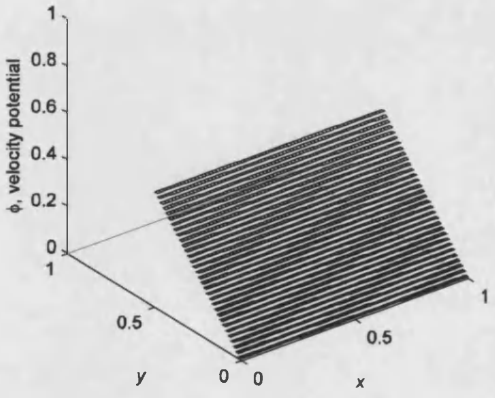


Figure 5.2. Standard numerical solution. Each cell is plotted at a height equal to the value of ϕ at its centre.

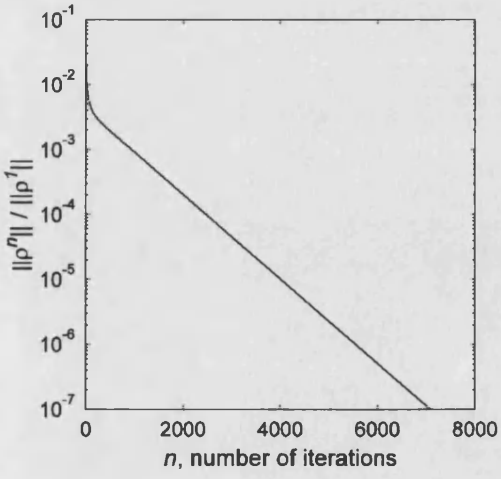


Figure 5.3. Reduction of residual norm of standard SOR solution.

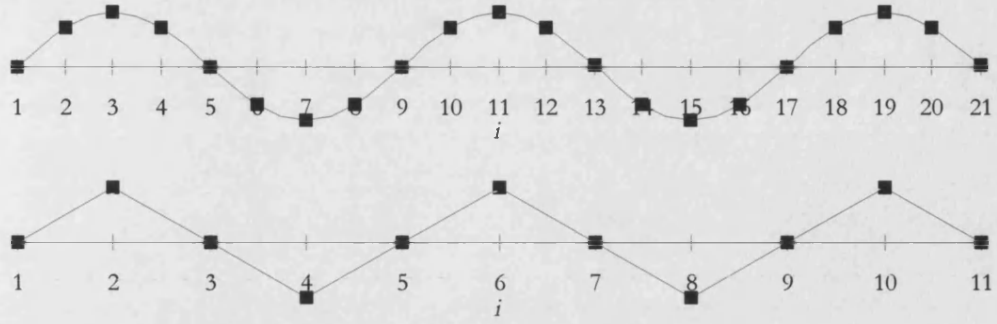


Figure 5.4. One-dimensional example of how an error field (black squares) is perceived as more oscillatory on a coarse grid than on a fine grid.

Figure 5.2. Figure 5.3 displays the reduction of the residual norm during the 7106 iterations required to satisfy equation (5.8). It can be observed that the error is quickly reduced in the early stages of the solution, after which the convergence rate slows down to a flat and constant logarithmic slope. This occurs because the Gauss-Seidel solver is quite efficient in reducing the oscillatory errors that occur during the early stages of the solution. However, once the error is sufficiently smoothed the solver has more difficulty in keeping a fast convergence rate.

5.2. Coarse Grid Correction Scheme

To overcome this inefficiency of the algorithm, multigrid iterations have been implemented. The multigrid strategy, initially suggested by Fedorenko (1964) and implemented by Brandt (1977), uses the fact that a relatively smooth error on a fine grid becomes more oscillatory once it is transferred to a coarse grid. This point is illustrated in Figure 5.4. The method is explained by the coarse grid correction scheme, which can be split into the following steps:

1. Smoothing of the error field by performing n_r iterations on the fine grid solution $\mathbf{A}_f\{\phi_f\}=\mathbf{q}_f$ using an arbitrary initial guess for ϕ (typically $\phi=0$) followed by computation of the residual from $\mathbf{?}_f^{n_r}=\mathbf{A}\{\phi_f\}^{n_r}-\mathbf{q}_f$;
2. Transfer of the fine grid residual vector to the coarse grid c using

$$\mathbf{?}_c = R\left(\mathbf{?}_f^{n_r}\right), \tag{5.9}$$

where R is an operator representing the transfer procedure from the fine to the coarse grid, called the restriction operator.

3. Performing n_p iterations on the coarse grid residual equation $\mathbf{A}_c\{\varepsilon_c\}_c=\mathbf{?}_c$ using initial guess $\{\varepsilon_c\}_c=\mathbf{0}$, where $\mathbf{A}_c$ is the discretisation matrix for the coarse grid.
4. Transfer the error vector to the fine grid and update ϕ using

$$\{\phi\}^{n_r+1}=\{\phi\}^{n_r}+P\{\varepsilon_c\}_c, \tag{5.10}$$

where P is an operator representing the transfer procedure from the coarse to the fine grid and $\{\varepsilon_c\}_c$ is the value of the convergence error on the coarse grid. P is often called the prolongation or interpolation operator.

5. Solve $\mathbf{A}_f\{\phi_f\}=\mathbf{q}_f$ n_r times on the fine grid, with initial guess $\{\phi\}^{n_r+1}$, after which compute the residual from $\mathbf{?}_f^{n_r+n_r}=\mathbf{A}\{\phi_f\}^{n_r+n_r}-\mathbf{q}_f$.
6. Repeat steps 2-5 until convergence is satisfied.

Step 1 smoothes the error until convergence deteriorates (in fact, the method can be made adaptive by using convergence as a parameter, rather than a fixed number of iterations, n_r and n_p). The values of n_r and n_p are usually small (between 2 and 5 iterations) since the error is rapidly smoothed by the Gauss-Seidel solver. The residual is then transferred to a coarse grid,

and the residual equation (5.7) is more oscillatory and hence easier to smooth. Furthermore, in Step 3, computations are now carried out in a coarser grid which represents a much cheaper process. The result is transferred to the fine grid and the solution is updated and smoothed further to correct ϕ.

5.3. Multigrid Level Generation

One of the main advantages of using the multigrid strategy with quadtree grids is the fact that there is no need to create any additional cells. This is often an expensive procedure when coarsening an irregular grid. However, due to the hierarchical structure of quadtrees all coarse grid cells are created in the grid generation process, and their co-ordinates and cell neighbourhood are readily available from their reference numbers. Here, two different methods to coarsen quadtree grids are described.

5.3.1. Smallest Cell Coarsening

The first method, used by Gáspár & Simbierowicz (1992), is simple to implement. Using the complete generated quadtree as the finest grid in the

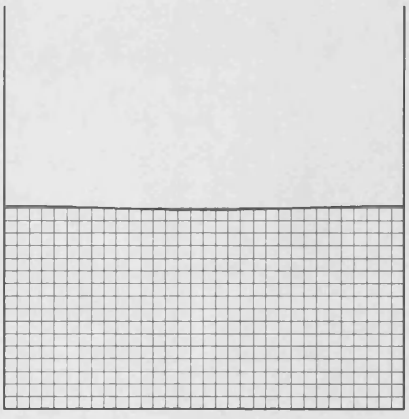


Figure 5.5. Coarser grid produced by pruning the highest division level of the grid in Figure 5.1

multigrid cycle, the coarser grids are automatically obtained by replacing the smallest cells with their larger ancestors. This is equivalent to successively pruning the highest division level from the quadtree. As an example, the coarser grid than that of Figure 5.1 is shown in Figure 5.5. The process is repeated for each of the desired multigrid levels. It follows that there is no need to create new grid cells, as they already exist in the overall quadtree. The algorithm simply changes which of the existing grid cells are selected for computation at each multigrid level by toggling the logical variable that identifies a cell as a leaf.

5.3.2. Global Coarsening

For a regular grid such as that in Figure 5.1, pruning the highest division level produces a grid which is coarser throughout the computational domain, since all cells are the same size. This conforms to the multigrid strategy and yields maximum convergence acceleration. However, for a quadtree grid such as that shown in Figure 5.6a, pruning the highest level produces localised coarsening restricted to regions of high grid refinement, illustrated in Figures 5.6b and 5.6c.

Since cells in the interior of the domain are unchanged the acceleration obtained from the multigrid method will not be as great as if coarsening was extended throughout the computational grid. To overcome this, an alternative global coarsening (GC) procedure is proposed here. Instead of being restricted to the smallest cells in the grid, the pruning procedure is extended to all cells by stipulating that any cluster of four sibling leaf cells are eligible for coarsening. Although the criterion is simple and straightforward, its implementation is not totally trivial because the coarsening procedure may increase the size difference between neighbouring cells beyond the maximum

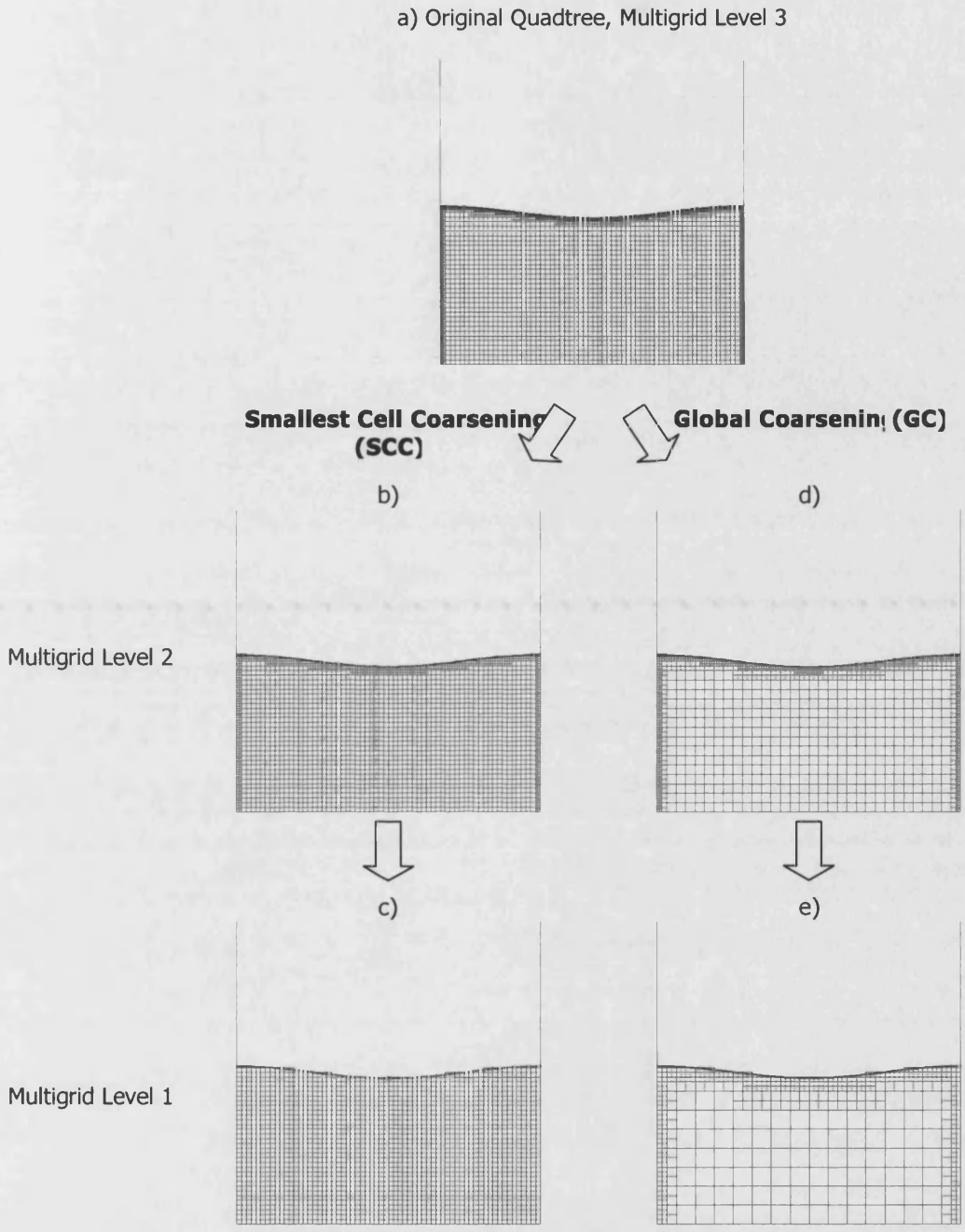


Figure 5.6. Two-level coarsening of quadtree a) using two different schemes: b) & c) coarsening of smallest cells; d) & e) global coarsening

one division level criterion. To ensure this is not violated, a grid normalisation routine must be checked every time cells are coarsened. Figure 5.6c demonstrates that this coarsening procedure produces a much more uniform grid coarsening, which should yield higher computational savings from the multigrid approach. These are expected to offset the slightly higher cost of the

global coarsening procedure with respect to the simpler method of pruning the highest division level.

Apart from the simplicity of generating coarser multigrid levels, quadtree grids offer other important advantages when coupled with a multigrid strategy. The first and more significant of these is the fact that the numbering system used to reference quadtree cells provides a ready link between each of the grids and, as a result, the need to keep link lists relating cells of different multigrid levels is avoided.

Secondly, since discretisation matrices are not stored in the present scheme, the memory costs of storing discretisation matrices for each multigrid level are spared. Instead, the algorithm simply selects the appropriate discretisation expression according to the neighbourhood of the coarse grid cells, as previously explained in Chapter 3. There is therefore no difference computationally between solving equations (5.7) and (5.2) and the same routines are used. The only modification concerns the boundary conditions which must be changed when solving the residual equation (5.7) on the coarser grid as follows. Since on the fine grid, the values of the velocity potential are prescribed at the free surface via a Dirichlet boundary condition, the equivalent free surface Dirichlet condition in the solution of the residual equation is $\varepsilon_c = 0$, i.e. there is no convergence error at the free surface since the value of ϕ is actually known. At rigid boundaries, where Neumann-type conditions $\partial\phi/\partial n = f$ are imposed, these are replaced by $\partial\varepsilon_c/\partial n = 0$ when solving the residual equation, since the convergence error at boundary cells is the same as that at the corresponding boundary points.

Identification of surface and non-submerged cells for the coarse multigrid levels is performed using the same subroutine described in Chapter

3 for the complete quadtree, with no need to coarsen or in any way manipulate the Lagrangian chain of particles defining the free surface curve.

Finally, the procedures to transfer information between fine and coarse grids are considerably simplified due to the regular topography of quadtree cells. These transfer procedures will now be introduced.

5.4. Grid Transfer Procedures

RESTRICTION OPERATOR

A general way of transferring a variable from the fine to the coarse grid is to compute the gradient at each of the fine cells and use this to extrapolate to the coarse cell centre. This is performed for each of the fine cells contained in the coarse cell and an average is then taken. Mathematically, this is expressed by

$$\rho_I = \frac{1}{n_f} \sum_{i=1}^{n_f} \left(\rho_i + (\nabla \rho)_i \cdot (\mathbf{x}_I - \mathbf{x}_i) \right) \tag{5.11}$$

where ρ_i are the nodal values of the residual (or any other variable), the roman numeral denotes the coarse cell, i is the index of fine grid cells, as illustrated in Figure 5.7a., n_f is the number of fine cells used in the restriction procedure and $\mathbf{x}$ is the positional vector of a cell. When a restricted topology is used, such as the quadrilateral cells used in this work, equation (5.11) reduces to the bilinear interpolation

$$\rho_I = \frac{\rho_1 + \rho_2 + \rho_3 + \rho_4}{4}. \tag{5.12}$$

At the surface, the situation may arise where one or more of the fine grid cells are not submerged and thus are not computational cells. When this is the

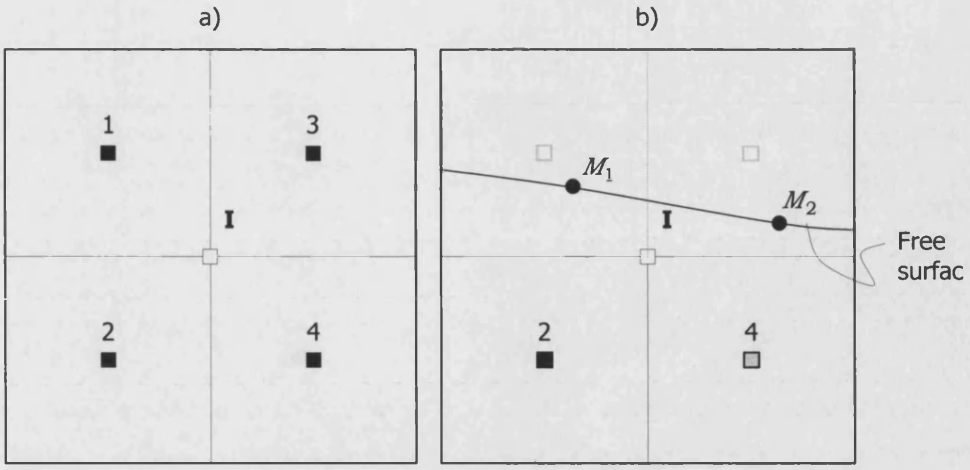


Figure 5.7 Solution transfer from fine (■) to coarse grid cells (□): a) restriction of interior cells; b) restriction near the surface when non-submerged cells (■) are present

case, these cells are excluded from the restriction procedure. Accordingly, for the arrangement depicted in Figure 5.7b, equation (5.12) is replaced by,

$$\rho_I = \frac{\rho_2 + \rho_4}{2}. \tag{5.13}$$

This was found to be sufficiently accurate and hence preferable to the full application of equation (5.11) which would have required the computation of the gradient of ρ at the fine cells and surface markers M_1 and M_2.

As Figure 5.6 testifies, there are also occasions where the same cell is a leaf cell on two multigrid levels, an occurrence that is especially frequent when the SCC method is used. When such cells are considered, the residual computed on the fine grid is simply injected unaltered onto the coarser grid.

PROLONGATION OPERATOR

Similarly, the interpolation of coarse grid data onto fine grid cells can be generally expressed by

$$(\varepsilon_c)_i = (\varepsilon_c)_I + (\nabla \varepsilon_c)_I \cdot (\mathbf{x}_i - \mathbf{x}_I), \tag{5.14}$$

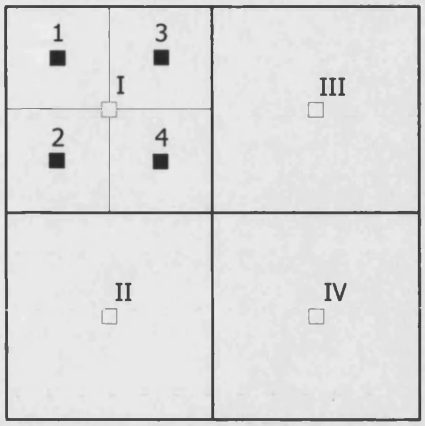


Figure 5.8. **Solution transfer from coarse(□) to fine grid cells (■)**

where $(\varepsilon_c)_i$ are the nodal values of the convergence error (or any other variable). Again, the use of a fixed cell topology allows the prolongation procedure to be simplified, so that the additional computation of the gradient of the error need not be performed.

Two prolongation operators were considered for the transfer of the convergence error from coarse to fine grids. The first is a zero-order procedure by which the value of the error on the fine grid cells is set equal to that at the corresponding coarse cell. Accordingly, the potential at the four smaller cells in Figure 5.8 is set equal to that of cell I, i.e.

$$(\varepsilon_c)_{1,2,3,4} = (\varepsilon_c)_I . \tag{5.15}$$

where $(\varepsilon_c)_i$ are the nodal values of the convergence error, and the subscripts refer to the cell arrangement of Figure 5.8.

Additionally, a first-order operator was devised by which the solution of the residual equation is transferred from coarse to finer grids using,

$$(\varepsilon_c)_1 = \frac{9(\varepsilon_c)_I + 3(\varepsilon_c)_{II} + 3(\varepsilon_c)_{III} + (\varepsilon_c)_{IV}}{16}, \tag{5.16}$$

The convergence error is then added to the existing solution of ϕ on the fine grid. Equation (5.16) is obtained by discretising Laplace's equation at node 1 for the distribution of coarse grid cells shown, so that the convergence error is mapped accurately onto the fine grid. Other restriction and prolongation operators can be employed, some of which are found in Hackbusch (1985), Briggs (1987); Hackbush (1985) and Wesseling (1988).

5.5. Multigrid Cycles

It can be noted that the coarse grid problem, that of solving the residual equation, is not much different from the original problem. Hence, the method described in Section 5.2 can be applied again to the residual equation using an even coarser grid. Recursively, the coarse grid correction scheme can be applied until the coarsest possible grid has been reached. The solution on the coarsest grid is then successively prolonged down to the original grid. The procedure is given next. The grids are now identified using the multigrid level to which they belong. The maximum multigrid level, L, is that of the finest grid, whilst the coarsest grid is used at the minimum multigrid level 1.

1. Solve $\mathbf{A}_L\phi_L = \mathbf{q}_L$ n_r times with arbitrary initial guess for ϕ.
2. Compute $\rho_{L-1} = \mathbf{R}\rho_L$
3. Solve $\mathbf{A}_{L-1}(\varepsilon_c)_{L-1} = \rho_{L-1}$ n_r times with initial guess $(\varepsilon_c)_{L-1} = \mathbf{0}$
4. Compute $\rho_{L-2} = \mathbf{R}\rho_{L-1}$
5. Solve $\mathbf{A}_{L-2}(\varepsilon_c)_{L-2} = \rho_{L-2}$ n_r times with initial guess $(\varepsilon_c)_{L-2} = \mathbf{0}$
6. Compute $\rho_{L-3} = \mathbf{R}\rho_{L-2}$
7. ⋮
8. Solve $\mathbf{A}_1(\varepsilon_c)_1 = \rho_1$ n_p times with initial guess $(\varepsilon_c)_1 = \mathbf{0}$
9. ⋮

10. Update $(\varepsilon_c)_{L-2} \leftarrow (\varepsilon_c)_{L-2} + \mathbf{P}(\varepsilon_c)_{L-3}$

11. Solve $\mathbf{A}_{L-2}(\varepsilon_c)_{L-2} = \rho_{L-2}$ n_p times with initial guess $(\varepsilon_c)_{L-2}$

12. Update $(\varepsilon_c)_{L-1} \leftarrow (\varepsilon_c)_{L-1} + \mathbf{P}(\varepsilon_c)_{L-2}$

13. Solve $\mathbf{A}_{L-1}(\varepsilon_c)_{L-1} = \rho_{L-1}$ n_p times with initial guess $(\varepsilon_c)_{L-1}$

14. Update $\phi_L \leftarrow \phi_L + \mathbf{P}(\varepsilon_c)_{L-1}$

15. Return to point 1.

The arrow notation represents computational assignment, i.e. replacement or over-writing. Figure 5.9 shows the schedule for four multigrid levels in the order with which they are visited for the multigrid cycle. Because of its pattern, the cycle described above is called the V-Cycle.

Using the test case of Figure 5.1, multigrid iterations using four multigrid levels are used to obtain a solution. At the maximum multigrid level, L = 4, the grid of Figure 5.2 is used, generated by setting $l_m = l_b = 6$, whilst at levels 3, 2 and 1, the grids of Figures 5.5, 5.10 and 5.11 are used. The number of iterations performed before restriction and prolongation are set respectively to $n_r = 3$ and $n_p = 3$. The multigrid solution thus obtained is shown in Figure 5.12 and can be seen to be identical to that obtained using the standard solver, displayed in Figure 5.3. However, when plots of the evolution of the update error during the iterative process are superimposed for

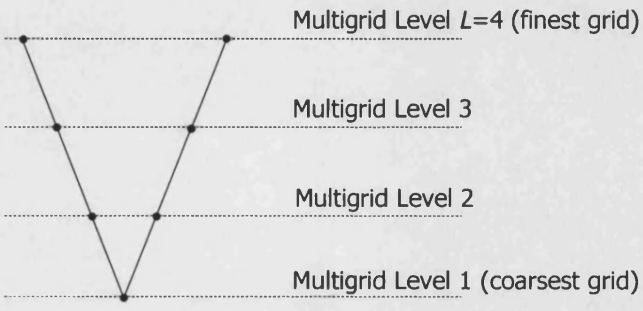


Figure 5.9 Grid visiting schedule for the multigrid V-Cycle

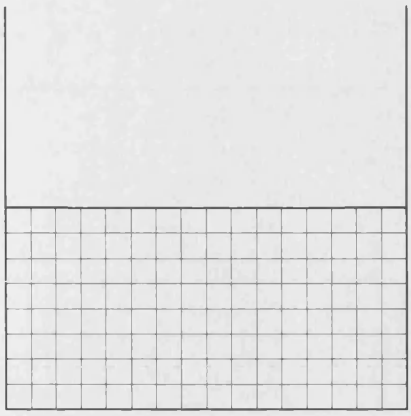


Figure 5.10. Multigrid level L=2

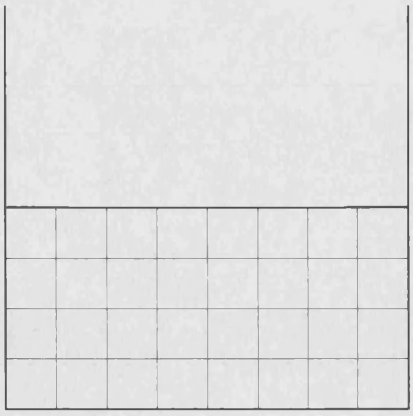


Figure 5.11. Multigrid level L=1

the standard and multigrid solutions, as displayed in Figure 5.13, it can be observed that, the convergence rate does not significantly decrease as the error is reduced. It can thus be observed that the employment of a multigrid strategy reduces the number of iterations on the fine grid required to satisfy the converge criterion from 7104 to only 120. This is equivalent to performing 40 multigrid V-cycles.

A complete analysis of the performance of the multigrid strategy used in conjunction with quadtree grids in terms of computational costs is included in

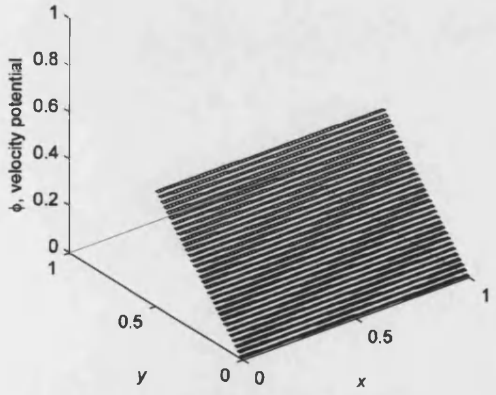


Figure 5.12. Multigrid numerical solution. Each cell is plotted at a height equal to the value of ϕ at its centre.

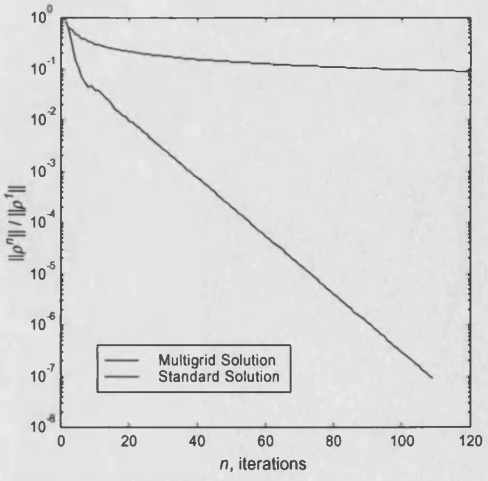


Figure 5.13. Reduction of residual norm of multigrid solver.

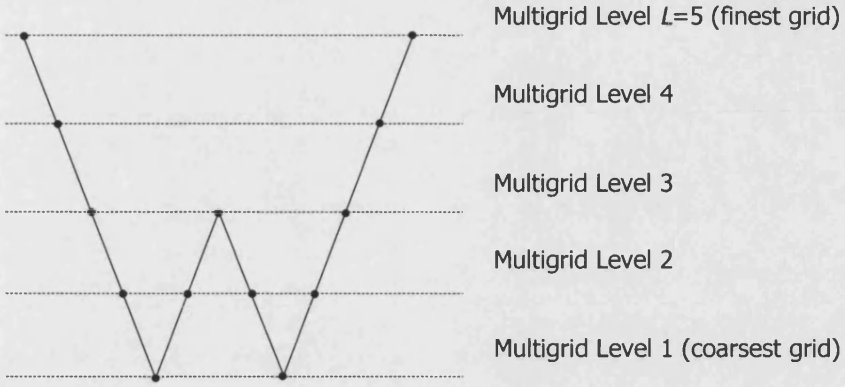


Figure 5.14 Grid visiting schedule for the multigrid W-Cycle

the next chapter.

An additional multigrid cycle is investigated in this work. It is called the W-Cycle, as the grid visiting schedule of Figure 5.14 suggests. As can be seen, additional work is performed on coarser grids before prolonging the solution up to the finest grid at the end of the cycle. It is used to investigate if the improved quality in the solution due to this additional work can offset the additional costs incurred by the additional computations on the coarser grids. The following chapter contains the results of these investigations.

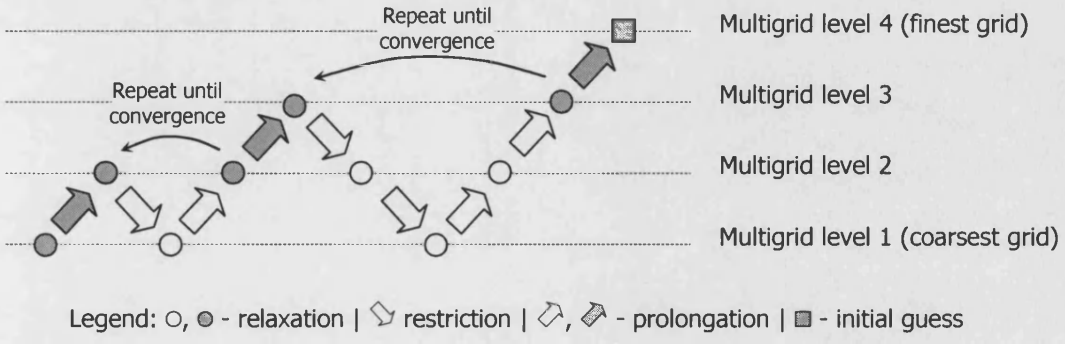


Figure 5.15. The full multigrid V-Cycle used to obtain a good initial guess.
Operations where the velocity potential is the independent variable are shaded grey.

5.6. Initial Guess Using Multigrid Iterations

The multigrid strategy can also be used to provide the solver (be it a multigrid or a standard one) with a good initial guess. The idea is to solve the governing equation on the coarsest grid and successively prolong it on to the finer grids, relaxing the solution until convergence after each prolongation. Once the solution reaches the finest quadtree, it should be accurate enough so that the additional work required to smooth it is much cheaper than if the standard $\phi=0$ guess is used. It follows that the prolongation operators of equations (5.14)-(5.16) are in this case applied to ϕ instead of ε_c. However, following the ideas of the multigrid strategy, this scheme can be improved if multigrid cycles are performed at each multigrid level. Schematically, this can be represented by Figure 5.15, using V-cycles as the example. The result is the full multigrid V-cycle (FMG). Once the initial guess is obtained at the end of the FMG cycle, the solution process can be carried out using the standard or multigrid solvers.

Figure 5.16a displays the initial guess obtained using the FMG scheme after it is interpolated onto the finest grid, prior to the solution process starts. The first-order prolongation operator of equation (5.16) is used in this case. The initial guess can be seen to be very similar to the converged solution of Figure 5.12. This is confirmed by the total error field depicted in Figure 5.16b, with mean total error being equal to just 1.25×10^{-4} or 0.19% of the exact solution. The standard solution using this initial guess is predictably more economical than that obtained using the initial guess $\phi=0$. Figure 5.17 displays the convergence histories of these two cases using a standard solver.

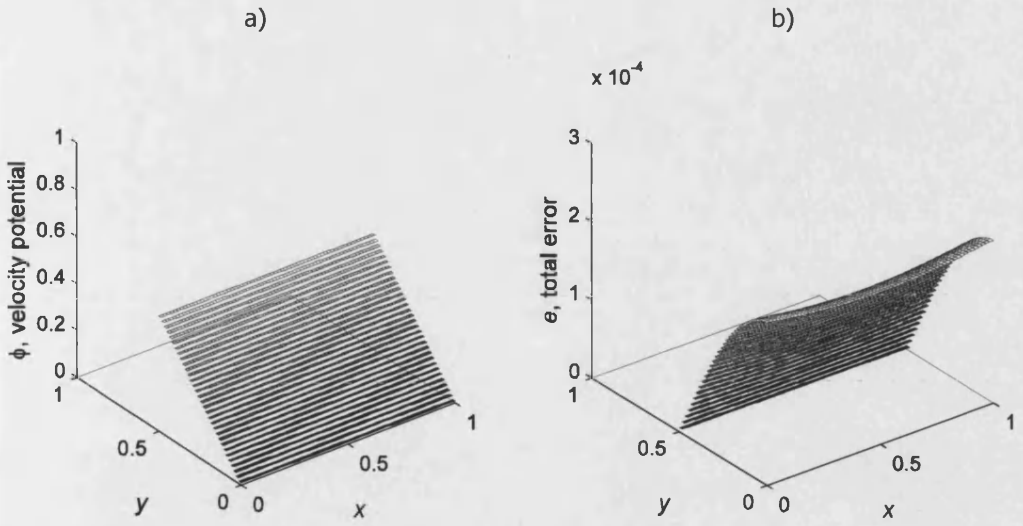


Figure 5.16.a) Initial guess to case of Figure 5.1 obtained using FMG and b) the total error of this guess (mean error equal to 1.25×10^{-4})

The convergence rate is maintained but, since the FMG solution starts at a much lower level of error it satisfies the convergence error using 800 less iterations (10% less). This is equivalent to savings in computational time of the order of 7%, the reduction in percentage being a cause of the CPU time required to generate the multigrid levels. When the multigrid solver is used, instead of the standard solver, these savings are increased to about 12% since

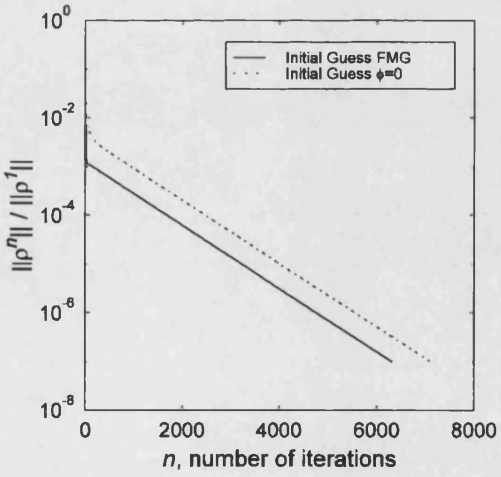


Figure 5.17. Convergence histories for the standard solver using two different initial guesses of the potential field.

the cost of generating the additional multigrid levels is incurred regardless of the initial guess used. That is, since multigrid levels are used in the solution, they may as well be employed in providing a FMG initial guess without any additional cost in grid generation procedures.

6. Results

In this chapter, numerical results are presented. It is divided into three sections. The first section deals with the validation of the code. With this in mind, the grid convergence of the code is investigated with respect to both the spatial and temporal discretisation. Comparisons between different time stepping schemes are also established. Comparisons are made with approximate theoretical solutions and published numerical data.

The second section deals with the performance of the multigrid method, in terms of increased convergence and consequent computational savings. The two techniques for the generation of the multigrid levels are compared, along with other multigrid parameters. The effects of multigrid acceleration are measured in terms of computational time and quality of the solution.

The third section is concerned with the simulation of nonlinear standing waves of varying amplitudes and water depths, and the effect of these parameters on the wave motion.

6.1. Grid Convergence

In this section, the grid convergence behaviour of the code is investigated, to establish that the code is consistent (i.e. errors are reduced when the spatial grid is refined), that it is stable (i.e. errors do not increase in the course of the solution) and that, as a result of satisfying the two previous conditions, the numerical method is convergent.

The validation of the code can initially be performed by simulating a standing wave of very small amplitude. When the wave amplitude is very small, there is little movement of the fluid in the computational domain. As a result, the non-linear term in the dynamic boundary condition (4.20) representing the kinetic energy at the free surface is much smaller than the potential energy term and bears practically no effect on the wave motion. If this is the case, the free surface dynamic boundary condition of equation (4.18) is linearized by,

$$gy+\frac{\partial\Phi}{\partial t}=0, \qquad \text{on } y=\eta, \tag{6.1}$$

where η is the wave elevation. Similarly, the kinematic condition of equation (4.12) reduces to,

$$\frac{\partial\Phi}{\partial y}=\frac{\partial\eta}{\partial t}. \qquad \text{on } y=\eta. \tag{6.2}$$

Differentiation of equation (6.1) with respect to time and substitution of equation (6.2) yields

$$\frac{\partial^2\Phi}{\partial t^2}+g\frac{\partial\Phi}{\partial y}=0, \qquad \text{at } y=\eta. \tag{6.3}$$

The boundary conditions at rigid boundaries and the governing equation remain unchanged,

$$\frac{\partial \Phi}{\partial n}=0, \qquad \text{at } y=-h,\ x=0,\ x=b, \tag{6.4}$$

$$\nabla^2\Phi=0, \qquad \text{on } \Omega_d. \tag{6.5}$$

where h is the mean water depth and b is the width of the tank as defined in Figure 4.3, n is the direction normal to each rigid boundary and Ω_d is the fluid domain. The problem is an initial value problem whose initial conditions are expressed by,

$$\Phi\big|_{t=0}=\varphi \tag{6.6}$$

$$\eta\big|_{t=0}=-\frac{1}{g}\frac{\partial \Phi}{\partial t}\bigg|_{t=0}=\gamma \tag{6.7}$$

where $\varphi(x,y)$ is a potential profile prescribed at the free surface and $\gamma(x)$ describes the initial free surface shape. The harmonic solution to the above equations can be written as

$$\Phi=\sum_{m=1}^{\infty} F_m(t)\frac{\cosh(k_m(y+h))}{\cosh(k_m h)}\cos(k_m x), \tag{6.8}$$

where $k_m=m\pi/b$. Substitution of equation (6.8) into equation (6.3) yields,

$$F_m(t)=A_m\cos(\omega_m t)+B_m\sin(\omega_m t), \tag{6.9}$$

where

$$\omega_m=\sqrt{k_m g\tanh(k_m h)}. \tag{6.10}$$

Replacing equation (6.8) in the initial conditions (6.6) and (6.7) gives,

$$\begin{aligned} A_m&=\frac{2}{b}\int_0^b \varphi\cos(k_m x)dx \\ B_m&=-\frac{2g}{b\omega_m}\int_0^b \gamma\cos(k_m x)dx \end{aligned} \tag{6.11}$$

For the case of a sinusoidal standing wave, with initial conditions given by,

$$\Phi\big|_{t=0}=\varphi(x,y)=0, \qquad \text{and} \qquad \eta\big|_{t=0}=\gamma(x)=a\cos(2\pi x/\lambda), \tag{6.12}$$

the solution to equation (6.11) is

$$A_m = 0, \qquad B_2 = -ag/\omega_2 \text{ and } B_m = 0, \text{ for } m \neq 2. \tag{6.13}$$

This yields the well-known linear solution,

$$\eta(x,t) = a\cos(\omega_2 t)\cos(k_2 x). \tag{6.14}$$

Wu & Eatock Taylor (1994) used a Stokes perturbation expansion of the velocity the following second-order elevation,

$$\eta_2(x,t) = \frac{1}{8g}\left(2(\omega_2 a)^2\cos 2\omega t + \frac{a^2}{\omega_2^2}\left(k_2^2 g^2 + \omega_2^4\right) - \frac{a^2}{\omega_2^2}\left(k_2^2 g^2 + 3\omega_2^4\right)\cos\omega_4 t\right). \tag{6.15}$$

To permit true comparisons with other numerical data, results are often non-dimensionalised as follows,

$$L' = \frac{L}{h}, \qquad t' = t\sqrt{\frac{g}{h}}, \qquad \omega' = \omega\sqrt{\frac{h}{g}} \tag{6.16}$$

where L is any length, t is time, ω is the wave frequency and the prime denotes non-dimensional values. Additionally, the wave elevation is non-dimensionalised by dividing it by the wave amplitude.

6.1.1. Spatial Grid Convergence

The solution of the quasi-linear problem is tested using a wave of non-dimensional amplitude $a' = 0.01$ and non-dimensional wavelength $\lambda' = 2$. This wave is simulated with a series of regular grids shown in Figures 6.1-6.4 using a dimensional time step $\Delta t = 0.005s$. Grid 1 contains 512 grid cells and 32 surface markers, and is generated by setting the maximum, l_m, and minimum, l_b, division levels to 5. By setting $l_m = l_b = 6$, grid 2 is produced, made up of 2048 cells and using 64 surface markers. Grids 3 and 4 are each refined by an extra division level and contain, respectively, 8192 and 32768 cells. In grid 3 the free surface is defined by 128 markers whilst in grid 4 256 surface markers are used.

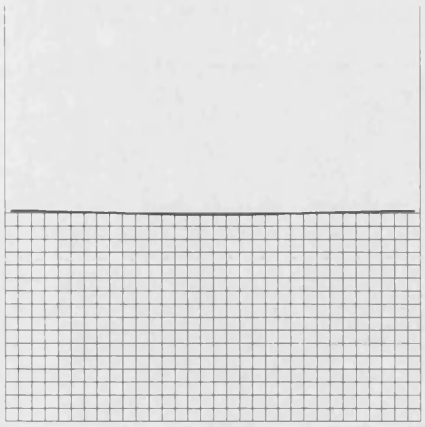


Figure 6.1. Grid 1, $l_m = l_b = 5$, 512 cells

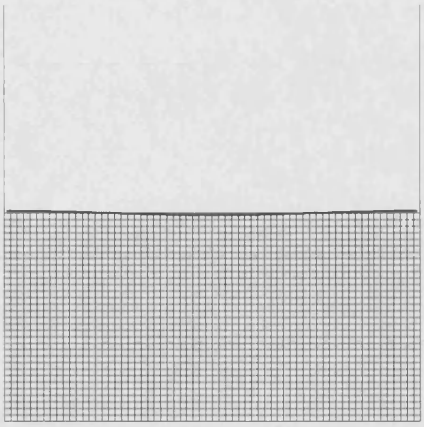


Figure 6.2. Grid 2, $l_m = l_b = 6$, 2048 cells

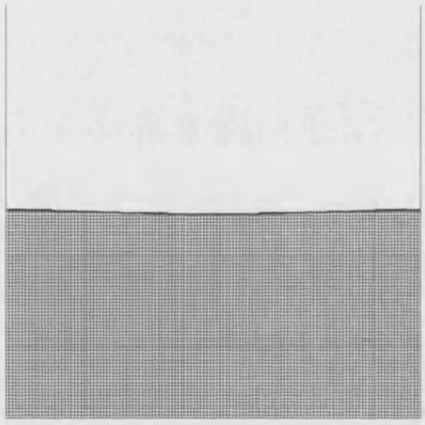


Figure 6.3. Grid 3, $l_m = l_b = 7$, 8192 cells

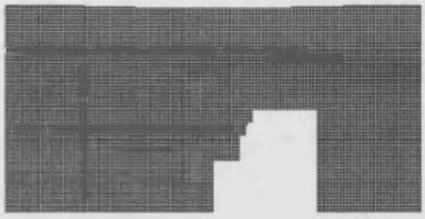


Figure 6.4. Grid 4, $l_m = l_b = 8$, 32768 cells

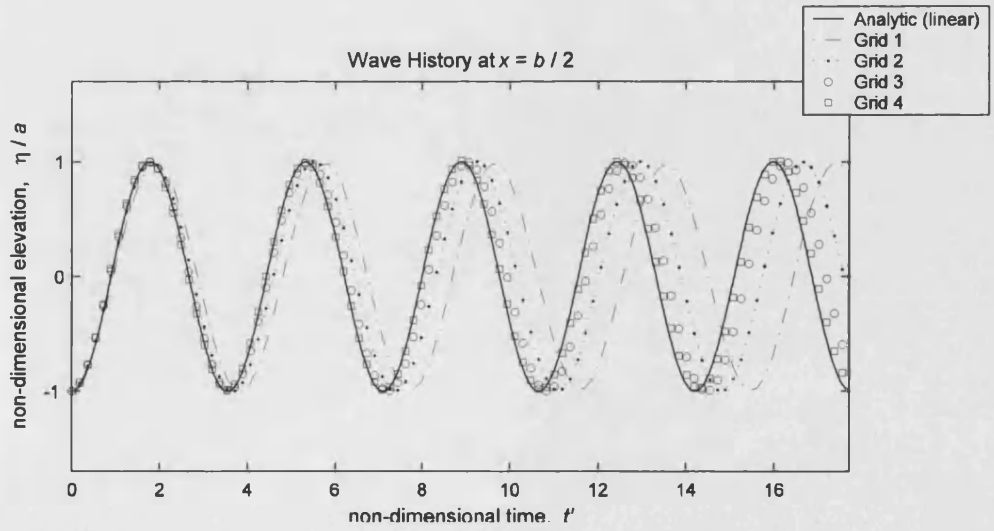


Figure 6.5. Time history of free surface elevations at the centre of the tank for four different regular grids.

The elevation histories at the centre of the tank obtained using each of these grids are collected in Figure 6.5. When the coarsest grid 1 is used, the time period is grossly overestimated by the numerical method. However, as the grid is refined, the wave period converges to the non-dimensional wave period $T'=3.5515$ predicted by linear theory. The numerical solution obtained using Grid 4 shows that for such small amplitude the linear theory is an accurate approximation to the fully non-linear behaviour.

Computations using such a large grid as Grid 4 are expensive due to the large number of cells: each time step requires on average 150.5s per time step. Simultaneously, the memory requirements such a large grid entails are of the order of 6.1 MB. The numerical method is susceptible to the quality of the spatial grid in two stages. The first is the computation of the velocity potential. The level of refinement across the whole domain affects the accuracy of the computed potential field. This is a consequence of the elliptic nature of Laplace's equation whereby discretisation errors affect the solution everywhere in the domain with varying degrees of intensity. Discretisation errors depend on the spacing between grid cells and the value of the third and higher derivatives of the velocity potential, as discussed in Chapter 3.

The second stage is the calculation of the free surface velocities. This is necessarily performed using a one-sided approximation in the region of the domain where the gradient and the curvature of the potential are highest. It can thus be speculated that a higher refinement near the free surface is advantageous for two reasons: the higher gradient and curvature in this region and the fact that fluid velocities must be calculated at the free surface. It is thus expected that the high resolution in the deeper areas of the fluid domain

does not significantly contribute to the overall accuracy of the free surface simulation.

To investigate this hypothesis, grid 5 in Figure 6.6 is used. This quadtree grid is generated using $l_m=8$ and $l_b=6$ which has the same resolution at the free surface as grid 4 but a coarser discretisation away from the free surface region. As a result, it contains a fraction of the number of cells as grid 4. However, the numerical solution obtained with this grid is comparable in accuracy to that of grid 4, as shown in Figure 6.8. Moreover, computational costs are reduced to 51.9s of CPU time per time step and 0.43 MB of storage. In fact, if the maximum division level is increased to 9, yielding grid 6 in Figure 6.7, the solution virtually coincides with linear theory.

6.1.2. Temporal Grid Convergence

The effect of time step size will now be discussed. All the wave elevation histories presented thus far have been calculated using a time step $\Delta t=0.005$ and the second-order Adams-Bashforth scheme of equation 4.37. It is found that, for the same time step size, the Euler scheme causes great divergence of the wave elevation history, as displayed in Figure 6.9.

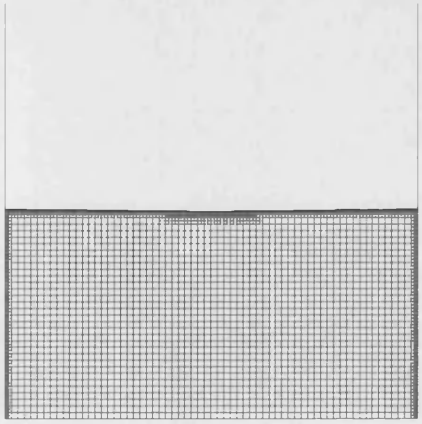


Figure 6.6. Grid 5, $l_m=8$, $l_b=6$, 3320 cells

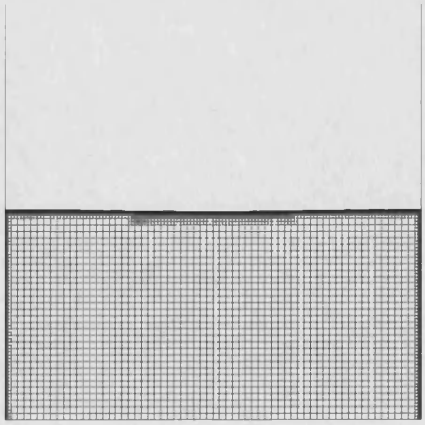


Figure 6.7. Grid 6, $l_m=9$, $l_b=6$, 5234 cells

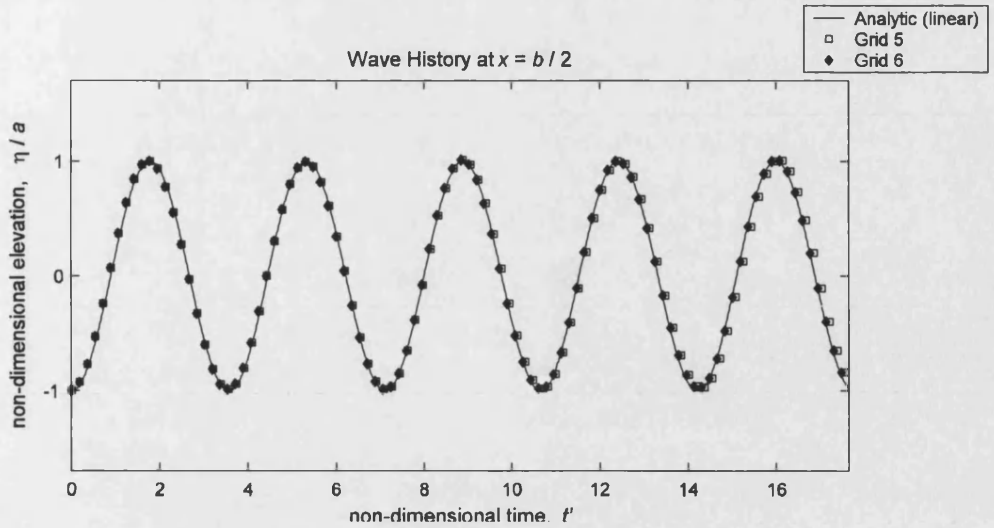


Figure 6.8. Time history of free surface elevations at the centre of the tank for a regular grid and two quadtree grids. $a' = 0.01$, $\Delta t' = 0.005$

Although a reduction in time step size by one order of magnitude greatly reduces this divergent behaviour, this arrives at considerable computational expense as the number of time steps increases by a factor of 10. Furthermore, some divergence is still observable as the simulation progresses and a further reduction in the time step does not eradicate this behaviour completely.

An increase in the order of the time integration scheme brings considerable improvements, and eliminates this divergent behaviour. This is

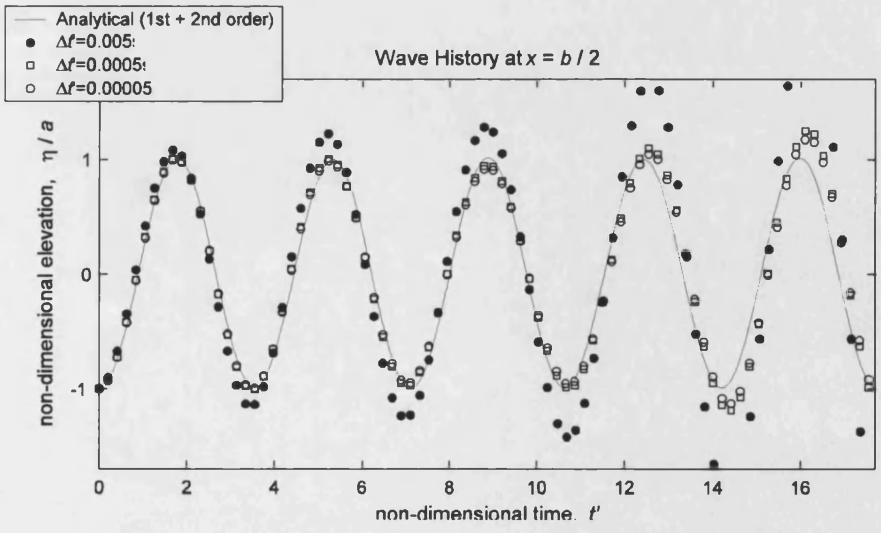


Figure 6.9 Divergent behaviour of the first-order Euler scheme for the case of Grid 5

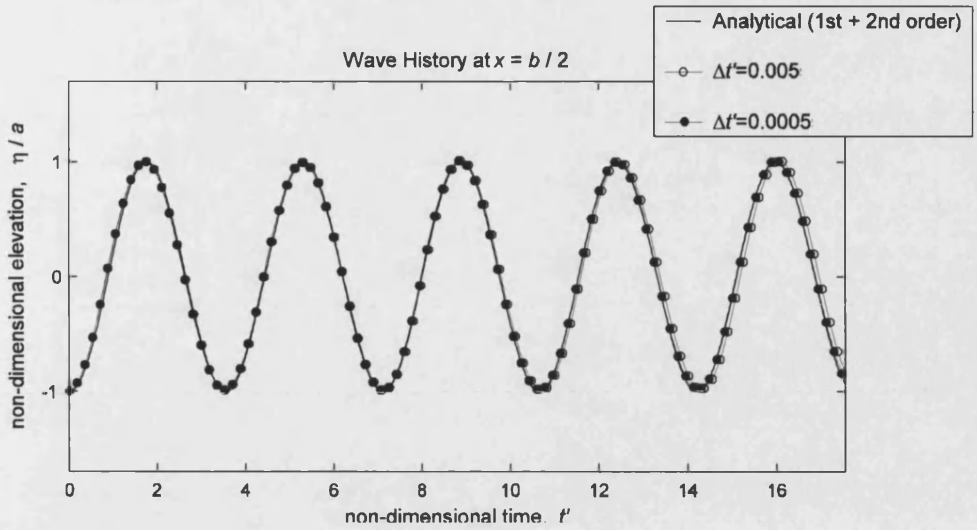


Figure 6.10 Time convergence for the Adams-Bashforth scheme.

further illustrated by the wave histories in Figure 6.10. It can be observed that even for the larger time step no divergence is observed. Furthermore, a further reduction in time step has virtually no effect on the solution. As a result, the Adams-Bashforth scheme with a time step of $\Delta t' = 0.005$ has been chosen for the remaining simulations in this chapter.

6.2. Performance of the Multigrid Method

An analysis of the apportionment of CPU time to each of the main components of the numerical method yields the pie chart of Figure 6.11. This was performed using the wave of Figure 6.6 at rest at $t = 0$. All values given in this Section 6.2 relate to this test case although the conclusions we draw have been confirmed using a variety of other test waves. Values of CPU time were averaged over 5 simulation of 100 time steps each. It shows that an overwhelming majority of computational time is spent in the solution of the potential field. This accounts for over 90% of the total CPU time. The only other significant task in terms of CPU time cost is the grid generation process and neighbour finding routines, accounting for around 3% of the cost. All

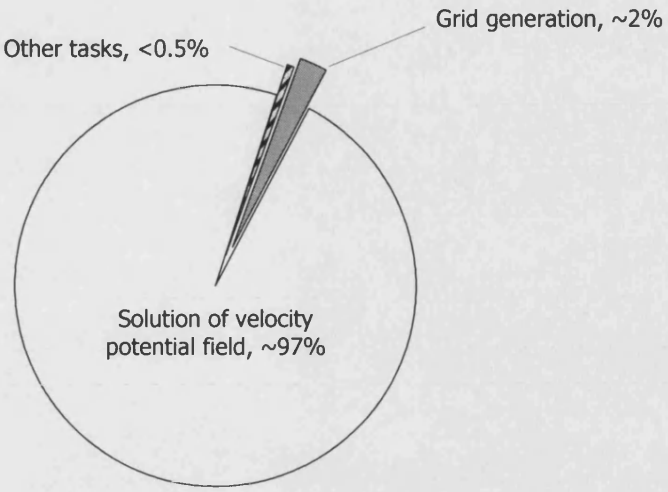


Figure 6.11. Relative computational cost per task for the standard solver (average values using Grid 5)

other tasks, which include the velocity calculation at the surface and the updating of free surface position and velocity potential, require less then 0.5% of the total time. Two complementary aspects can be supposed from this: that there is a large potential for improvement of the performance of the standard numerical method that solves the potential field; and that the grid generation process is an economic procedure.

It is thus clear that, whatever the improvement in performance of the solution of the ϕ field provided by multigrid acceleration, this will translate in its near entirety onto the overall performance of the method. Furthermore, the relative cheapness of the grid generation process augurs that the additional grid-related operations involving the various multigrid levels will be easily offset by any savings in the solution of Laplace's equation.

In the following sections, the performance of the multigrid strategy in the solution of the ϕ field will be analysed. The effect of five different parameters are investigated: 1) the coarsening scheme; 2) the order of the restriction and prolongation operators; 3) the number of multigrid levels; 4)

the number of iterations performed on each multigrid level; and 5) the visiting schedule of grids, i.e. the multigrid cycle type.

6.2.1. Comparison Between Coarsening Schemes

The two coarsening schemes discussed in Chapter 5 are applied to grid 5 to generate a number of coarser multigrid levels, as previously depicted in Figure 5.6. Results in terms of CPU time per time step are tabulated on Table 6.1. The multigrid levels are visited according to the V-cycle schedule, with 3 iterations performed at each level, i.e. $n_p = n_r = 3$. The first order restriction operator is used to transfer data from fine to coarse grids whilst the zero-order prolongation operator is used for the converse procedure. It is found that, when coarsening is restricted to the smallest cells in the grid (smallest cell coarsening, SCC method), the use of a three-level multigrid scheme increases the CPU time required to reach convergence by nearly 30%. This is due to the fact that the majority of computational cells are unchanged between multigrid levels, and as a result the frequency of the error is relatively unchanged between multgrid levels, and no significant advantage is gained in accelerating convergence. As a result, the additional costs of generating the coarser grids and transferring the solution between these are not offset by an increase in performance of the solver. By contrast, the global coarsening (GC) scheme increases cell size throughout the quadtree, which entails that the convergence error will be made more oscillatory on the coarser grids and, consequently, more quickly reduced. As a result, for a three-level multigrid V-cycle, the GC method affords a saving of 67.6% in computational time.

6.2.2. Number of Multigrid Levels

When more multigrid levels are used, the GC method continues to yield a better performance than that of the SCC grids but the difference between the two becomes less dramatic as more coarse grids are used. This is due to the fact that, after the SCC method coarsens the quadtree down to the minimum division level, $l_b = 6$, as is the case in Figure 5.6c, any further coarsening will be as global as that produced by the GC approach and good acceleration is then achieved. For both schemes, the solution process is accelerated by the use of more multigrid levels with an optimum performance being achieved when the global coarsening scheme is used to generate 6 multigrid levels. For this case, computational time is dramatically reduced by nearly 89.6% compared to the standard solution whilst, for the same number of multigrid levels, the GC scheme outperforms the SCC method by 12.9%. If a seventh level is employed, however, a slight deterioration in convergence is observed for both coarsening techniques, which is thought to be due to the excessive coarseness of this additional grid, composed of 8 cells only.

An analysis of the computational time allotment per task now shows

Table 6.1. Performance of multigrid V-cycle iterations using two grid coarserning schemes: smallest cell coarsening (SCC) and global coarsening (GC)

Standard Solver		Number of Multigrid Levels	**Multigrid SCC**			**Multigrid GC**			
				CPU time			CPU time		
Iterations	CPU time (s)		Iterations on finest grid	(s)	vs standard solution (%)	Iterations on finest grid	(s)	vs SCC solution (%)	vs standard solution (%)
		3	1633	67.5	+29.9	511	16.8	-75.1	-67.6
		4	689	37.3	-20.3	182	8.6	-76.9	-83.6
4154	51.9	5	204	13.5	-74.0	83	5.9	-56.3	-88.7
		6	57	6.2	-88.0	59	5.4	-12.9	-89.6
		7	66	6.9	-86.7	69	5.9	-14.5	-88.7

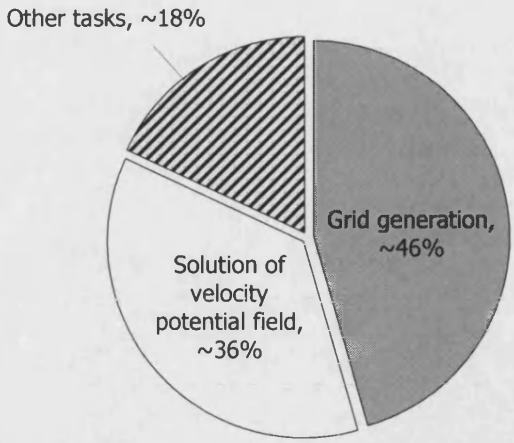


Figure 6.12. Relative computational cost per task for the GC multigrid solver with 6 multigrid levels (average values using Grid 5)

that a much more significant portion of the total cost is taken by grid related procedures, whose share has increased from 3% (cf. Figure 6.11) to 46%. Additionally, the proportion of total computational time spent performing the remaining tasks has increased to 18%, of which over 90% is taken up by the free surface velocity calculation, specifically by the discretisation of Laplace's equation at the interior points P used to calculate the component normal to the surface.

6.2.3. Grid Transfer Schemes

The effect of different grid transfer procedures on the efficiency of the multigrid acceleration is also of interest. In what concerns the restriction procedure of transferring the solution from fine to coarse grids, the first order averaging scheme described in Section 5.3.3 proved efficient at passing the high-frequency modes of the fine grid convergence error onto coarser grids. Higher order schemes using additional neighbouring cells were attempted but these introduced oscillation in the reduction of the residual error and, as a result, proved less efficient in accelerating convergence when irregular quadtree grids were employed.

Both prolongation schemes described in Chapter 5 are effective in accelerating convergence. Direct injection prolongation by which all fine grid cells are corrected using the convergence error of their coarser parent yields very good convergence. However, the first-order scheme, in which the convergence error is bilinearly interpolated from the coarse grid onto the fine grid cells, is marginally (approximately 5%) faster and is therefore chosen. All multigrid results presented are obtained using this scheme.

6.2.4. Number of Iterations on Each Multigrid Level

The effect of the number of iterations at each multigrid level was investigated by varying two parameters: the number of iterations prior to restriction, n_r, and the number of iterations prior to prolongation, n_p. Table 6.2 displays the results for a range of small values of n_r and n_p. The optimum setting is found to be $n_r = n_p = 3$. Reducing this value to 2, degrades convergence as the convergence error field is insufficiently smoothed after a grid transfer. If a higher number of iterations is used the convergence rate on each grid degrades slightly and the acceleration potential of the multigrid strategy is not fully explored.

Table 6.2. Effect of varying the number of iterations at each multigrid level, n_r and n_p. Results obtained using the GC multigrid V-cycle method on 6 multigrid levels for Grid 5.

n_r	n_p	Iterations on finest grid	CPU time	
			(s)	vs standard solution (%)
2	2	114	7.6	-85.3
3	3	59	5.4	-89.6
4	4	64	5.4	-89.6
5	5	106	6.5	-87.5
6	6	114	6.8	-86.9

Table 6.3. Effect of varying the number of iterations at each multigrid level, n_r and n_p, for the W-cycle. Results obtained using the GC multigrid method on 6 multigrid levels for Grid 5.

n_r	n_p	Iterations on finest grid	CPU time	
			(s)	vs standard solution (%)
3	3	60	5.9	-88.7
4	4	54	5.5	-89.5
5	5	57	5.6	-89.2

Combinations of values of n_r and n_p when $n_r \neq n_p$ afforded no further improvement on the optimal result shown in Table 6.2.

6.2.5. Visiting Schedule of Grids

The multigrid results presented so far have all been obtained using the V-cycle scheme (cf. Section 5.3.2) according to which the solution is successively restricted from the finest to the coarsest grid and subsequently interpolated back to the original grid. This is the most common cycle used in multigrid iterations but it is worth investigating if a change in the grid scheduling yields any further advantages.

For this purpose, the W-Cycle, whose schedule is graphically represented in Figure 5.14, was employed. Table 6.3 contains results for the performance of this cycle for a range of values of n_r and n_p. It is observed that the W-Cycle has a similar performance to that of the V-cycle. A reduction in the number of iterations on the finest grid is achieved but the extra calculations on the coarser multigrid levels causes a marginal increase in computational time. More interestingly, the optimum setting for the number of iterations at each multigrid level is higher than that found for the V-cycle, demonstrating the interdependence of these two parameters of the multigrid strategy.

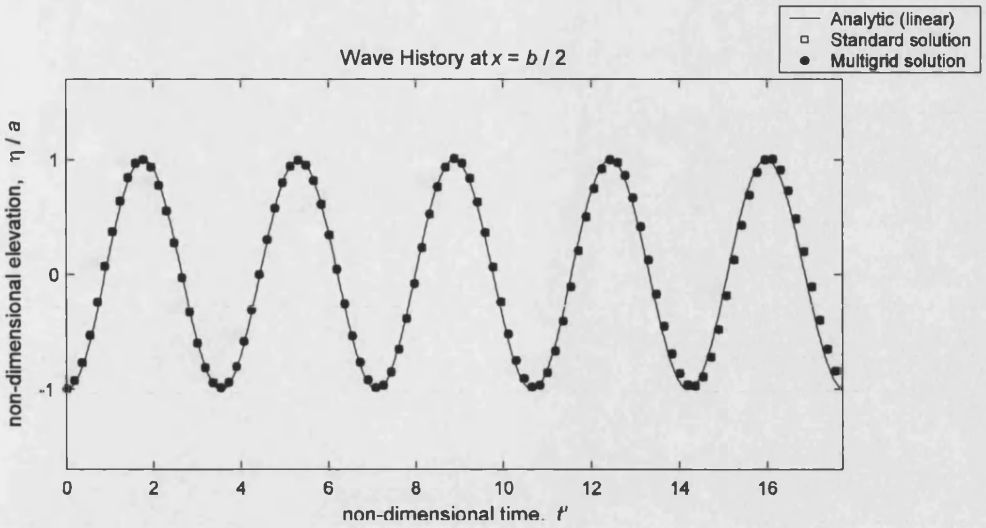


Figure 6.13. Comparison between wave histories for grid 5 obtained using the standard solver and the GC multigrid solver [6 level V-cycle, $n_c = n_p = 3$]

6.2.6. Multigrid Strategy to Obtain an Initial Guess

In Section 5.6, the use of the full multigrid (FMG) cycle to obtain an accurate initial guess was described using a sample case which satisfied the exact solution $\phi = y$. For such a case, improvements of the order of 12% were achieved. When a FMG initial guess is used at each time step of the simulation of non-linear water waves, the CPU time values obtained so far are further improved by approximately 5-7%. Accordingly, the optimum improvement setting found so far, i.e. that in Table 6.2, where the V-Cycle with $n_p = n_r = 3$ afforded a saving of 89.6% in computational time, yields a greater improvement of 91.2% when the FMG initial guess is used. As a result, FMG initial guesses are used in obtaining all the wave histories presented hereafter.

Finally, to complete the analysis of the multigrid strategy it is important to confirm that the multigrid and standard solutions coincide and that no difference is observable as the simulation time marches on. Figure 6.13 demonstrates that the two solutions are indistinguishable.

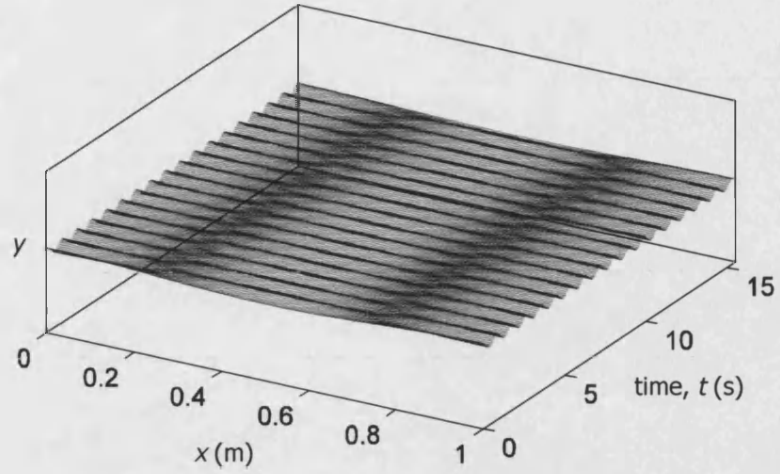


Figure 6.14. Evolution of wave of Figure 6.6 over 17 periods.
Sequence of surface profiles separated by 10 time steps of $\Delta t = 0.005$.

6.3. Conservation of Energy and Mass

As mentioned in Section 4.2, the main concern when the finite difference scheme is employed is that conservation is not explicitly enforced in the formulation. It is therefore of interest to investigate the evolution of the wave energy and mass during the numerical simulation.

A long-time simulation of the wave of Grid 5 is used to determine whether any instability in the wave profile is observed when a great number of time steps are employed. Figure 6.14 displays a succession of wave profiles, plotted at an interval of 10 time steps, encompassing the first 17 periods of motion. The simulation was carried out for 200 periods without change in the wave amplitude. This is used to demonstrate that the present method does not suffer from the numerical dissipation problems of other numerical methods which tend to smooth out the wave with progressing computational time. The change in wave energy (potential (PE) and kinetic (KE) is tracked during the simulation using the formulae,

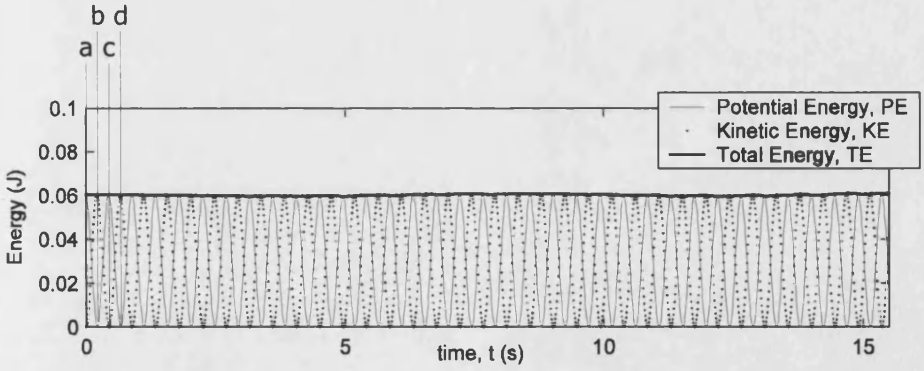


Figure 6.15. Evolution of potential, kinetic and total wave energies for the simulation of Figure 6.14.

$$KE = \frac{\rho}{2} \int_{\Omega_d} |\nabla \phi|^2 d\Omega_d \qquad \text{and} \qquad PE = \rho g \int_0^b \eta^2 dx, \tag{6.16}$$

where Ω_d is the fluid domain and $\rho = 1000\,\text{kg/m}^3$ is the density of water . It is assumed the domain has unit length in the third spatial dimension. The kinetic energy is computed by integrating the square of the fluid velocities over the fluid domain. This is performed in a post-processing stage using MATLAB built-in functions since internal velocities are not calculated during the simulation. The potential energy is numerically integrated over the free surface profile using quadratic polynomial fittings to chains of three surface markers, in a piecemeal fashion. Their sum represents the total wave energy $TE = PE + KE$. In Figure 6.15, the evolution of the three energy quantities is plotted over the simulation time of Figure 6.14. The four instances in time labelled (a-d) correspond to the plotted velocity potential and velocity fields of Figure 6.16. The kinetic and potential energies oscillate with a phase difference of π and a frequency twice that of the wave motion. As the wave is initiated from rest, the kinetic energy at $t = 0$ is zero, whilst the potential energy is at a maximum (Figure 6.16a). Once the wave is set in motion the potential energy is gradually converted into kinetic energy at the surface until the surface profile is flat (Figure 6.16b), at which point the potential energy is

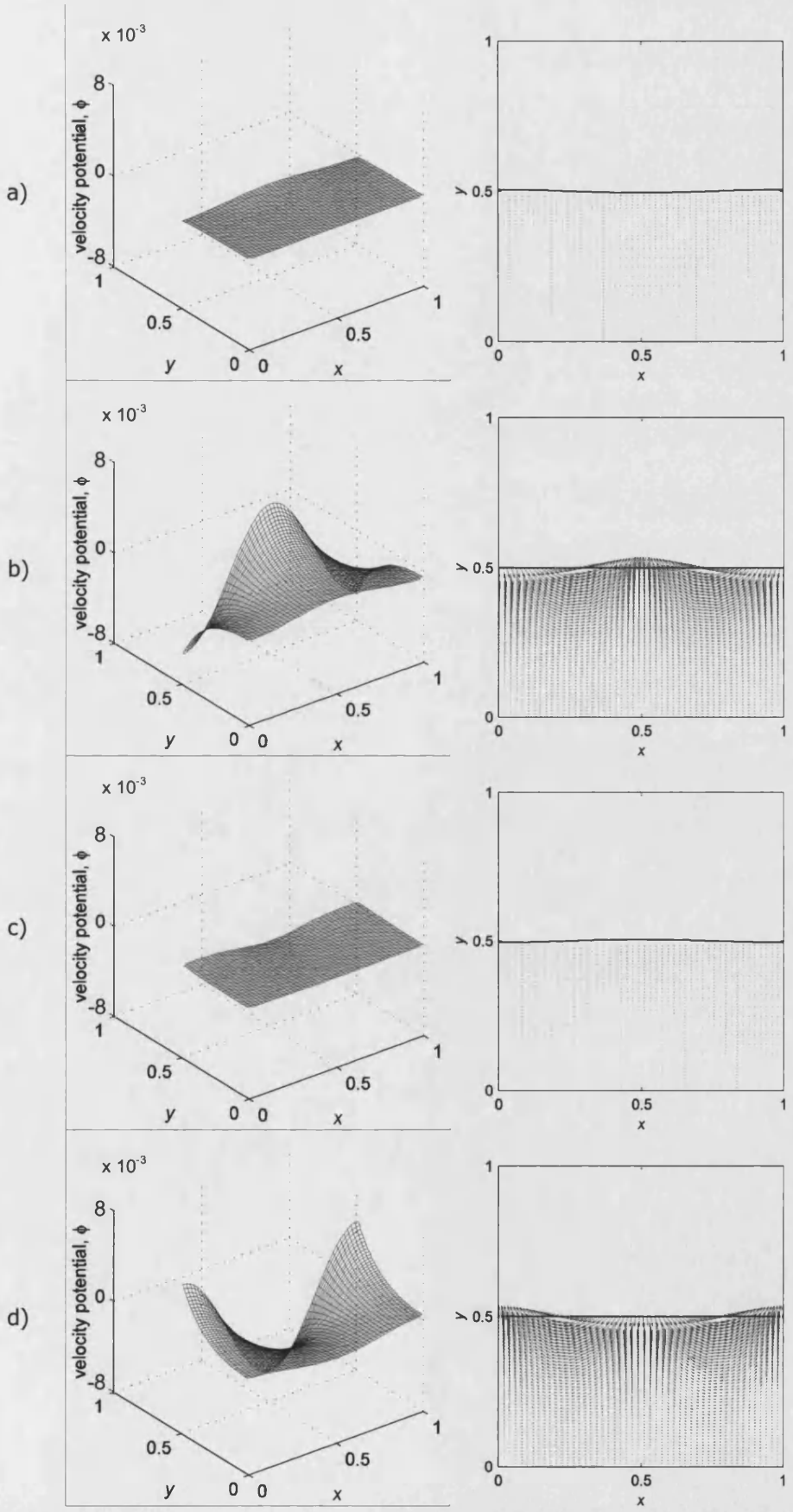


Figure 6.16. Velocity potential field and fluid velocities at every quarter wave period as labelled in Figure 6.15: a) $t=0$; b) $t=T/4$; c) $t=T/2$; d) $t=3T/4$

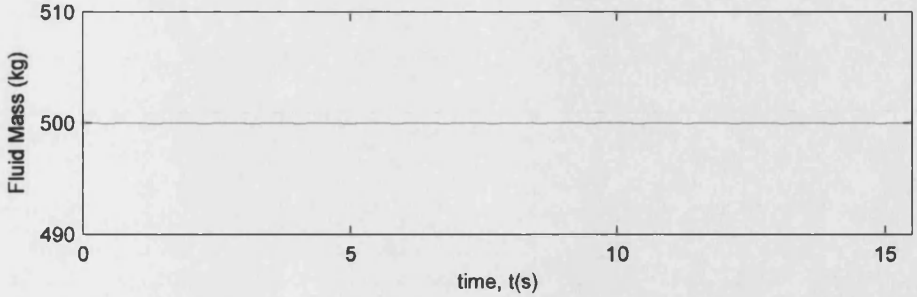


Figure 6.17 Evolution of fluid mass for the simulation of Figure 6.14.

zero and the wave energy is wholly kinetic. The motion proceeds until the wave comes to rest at its half-period, where the energy has been converted back into potential form (Figure 6.16c). The return path of the wave motion sees the wave being accelerated again to its maximum velocity which occurs when the profile is again flat (Figure 6.16d) until it returns to its position at rest as depicted in Figure 6.16a, completing one wave period.

The total energy curve of Figure 6.15 is seen to remain flat during the simulation, with small variations about a constant norm, indicating that energy is conserved by the scheme. Since there is no monotonic variation in the amount of total energy of the wave, these small oscillations are likely to be caused by small inaccuracies in the numerical integration of equation (4.16) rather than any lack of conservation of the numerical scheme.

The observance of conservation of mass of the incompressible fluid can be assessed by integrating the two-dimensional surface profile over the length of the numerical tank, to yield

$$m = \rho \int_0^b y dx . \tag{6.17}$$

In the computation of equation (6.17), the domain is taken to have unit length in the third spatial dimension. The integral in this equation should remain equal to the product of the mean water level by the breadth of the numerical tank, which for the quasi-linear wave of Figure 6.6 is $bh = 0.5$. The fluid mass

at each time step has been computed by numerical integration of equation (6.17) to yield the curve in Figure 6.17. No apparent change in the net fluid mass is observable. For the simulation shown, the fluid area was found to oscillate smoothly with the same frequency of the wave motion and a maximum deviation from the exact value of 0.003%. This very small variation was not found to increase with computational time, suggesting that it is more likely a result of the numerical error introduced by the numerical integration of equation (6.17) than any artificial lack of mass conservation by the solver. In Figures 5.16 and 5.17, mass and energy conservation are achieved over a total of 3100 time steps without any explicit enforcement of conservation in the numerical method. This is in contrast to some existing methods, which at each time step multiply the wave's y co-ordinates and velocity potential by a factor that ensures conservation of mass and energy.

6.4. Det Norske Veritas Study

In 1994, the Det Norske Veritas (DNV) Research Agency conducted a comparative study of fully non-linear wave simulation programs, details of which are provided by Nestegård (1994). The study included two test cases: a

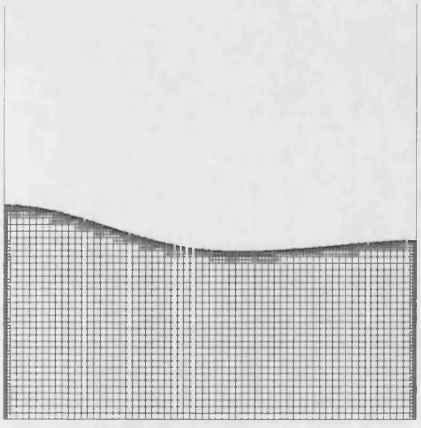


Figure 6.18. Asymmetric sloshing wave and associated mesh ($l_m = 8$, $l_b = 6$), at time $t = 0$

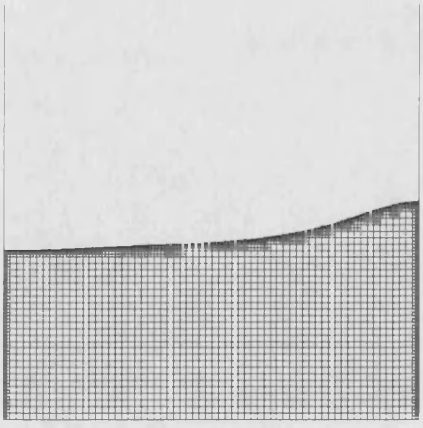


Figure 6.19. Sloshing wave and associated mesh at time $t = 9.2\,\mathrm{s}$

two-dimensional sloshing wave and the 3D diffraction of non-linear regular waves by a vertical surface piercing cylinder. It is the first of these that is used here to assess the performance of the current scheme. It consists of a two-dimensional sloshing wave at rest at $t=0\,\mathrm{s}$ with a mean water level $h=70\,\mathrm{m}$, in a tank of breadth $b=160\,\mathrm{m}$. The initial elevation profile of the wave is described by,

$$\eta(x,t=0)=\alpha\left[1-\left(\frac{x}{\beta}\right)^2\right]e^{-(x/\gamma)^2},$$

where $\alpha=12\,\mathrm{m}$, $\beta=53\,\mathrm{m}$ and $\gamma=76\,\mathrm{m}$. Figure 6.18 displays the wave at $t=0\,\mathrm{s}$ and the quadtree grid employed in the simulation, where $l_m=8$ and $l_b=5$. The different methods used in the DNV study were quantitatively assessed by comparing the computed values of elevation and velocities of the surface at co-ordinate $x=60\,\mathrm{m}$ and time $t=9.2\,\mathrm{s}$, corresponding to a non-dimensional time

Table 6.4. Results of surface elevation and velocities $x=60\,\mathrm{m}$ and $t=9.2\,\mathrm{s}$ from participants of the DNV comparative study (Nestegård (1994)) and the present method.

Elevation [m]	Horizontal Velocity u [m/s]	Vertical Velocity v [m/s]
-3.803	-2.456	-0.363
-3.860	-2.480	-0.560
-3.815	-2.423	-0.577
-3.759	-2.411	0.602
-3.820	-2.417	-0.580
-3.803	-2.417	-0.572
-3.720	-2.480	-0.690
-3.811	-2.411	-0.550
-3.810	-2.240	-0.560
-3.789	**-2.419**	**-0.441**

Present Method

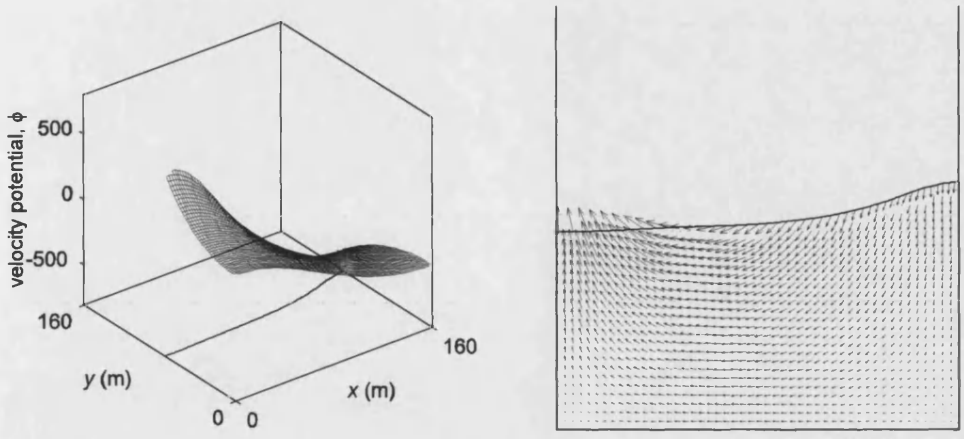


Figure 6.20. Velocity potential and velocity vector field at $t = 9.2$ s.

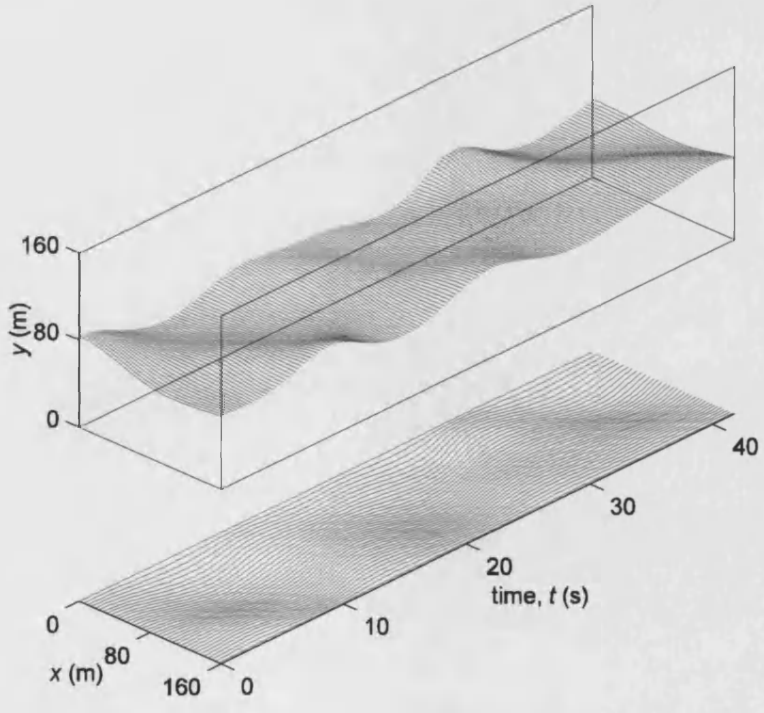


Figure 6.21. Motion of the sloshing wave of Figure 6.18.
The horizontal motion of surface markers is plotted at bottom of plot.

$t' = 3.443$. At this point in time, the surface profile was computed to be that shown in Figure 6.19, which shows good agreement with the published results of Greaves *et al.* (1997). The corresponding velocity potential and velocity vectors are displayed in Figure 6.20. Table 6.4 contains the numerical results of the elevation and velocities at the specified point in time and x-ordinate obtained by the participants in the DNV study (Nestegård (1994)). The values yielded by the present method show very good agreement.

Figure 6.21 displays the succession of surface profiles at intervals of 0.25s during the first 8500 time steps, where $\Delta t = 0.005$ s. At the bottom of the three-dimensional plot, the horizontal paths of selected surface markers are displayed (not all are included for visualisation purposes) to illustrate the considerable movement in the x-direction that the particles undergo.

6.5. Non-Linear Standing Waves

The need for a fully non-linear method is emphasised when waves of larger amplitude are investigated. In this case, non-linear effects have a strong influence on the wave behaviour and linear and second order theoretical predictions are inadequate to describe the true motion of the wave.

The shallow wave of Figure 6.22 has a non-dimensional amplitude $a' = a/h = 0.2$ and a non-dimensional wave length $\lambda' = \lambda/h = 10$. Its motion is simulated using the grid shown where $l_m = 8$, $l_b = 6$. The elevation history of such wave soon departs from the analytic predictions of linear and second-order theory, as displayed by the wave elevation history at the centre of the tank ($x = b/2 = 0.5$) in Figure 6.23. This agrees well with the numerical results of Greaves *et al.* (1997). The third trough in particular is less deep than the first two and exhibits extra deflections in the curvature of the history line.

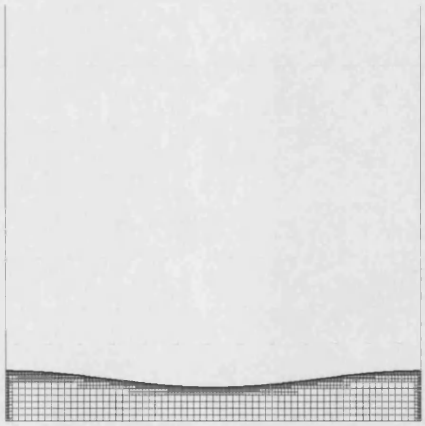


Figure 6.22. Shallow wave at $t=0$, $a'=0.2$, $\lambda'=10$

This appears to be a consequence of the shallowness of the wave which causes a reflection of the bottom of the tank, splitting the standing wave into two opposing sloshing waves. As a result, the surface profile develops the double crests seen crisscrossing the time-x surface in Figure 6.24.

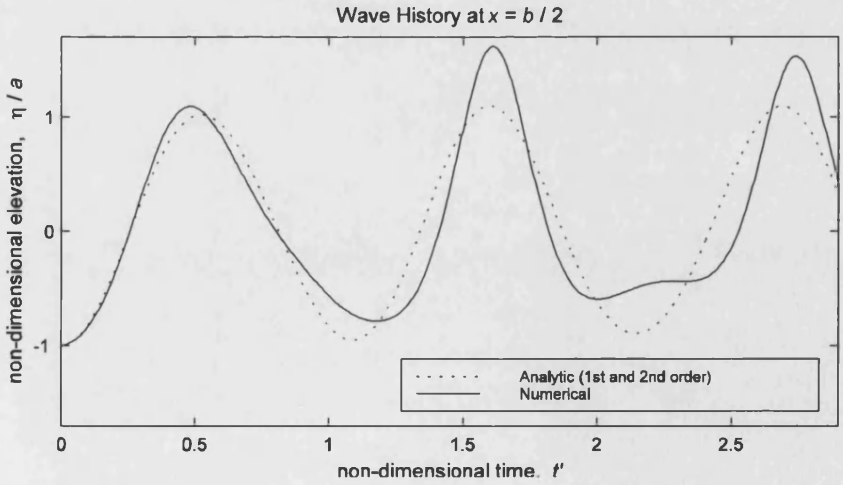


Figure 6.23. Wave elevation history at the centre of the tank of wave in Figure 6.22.

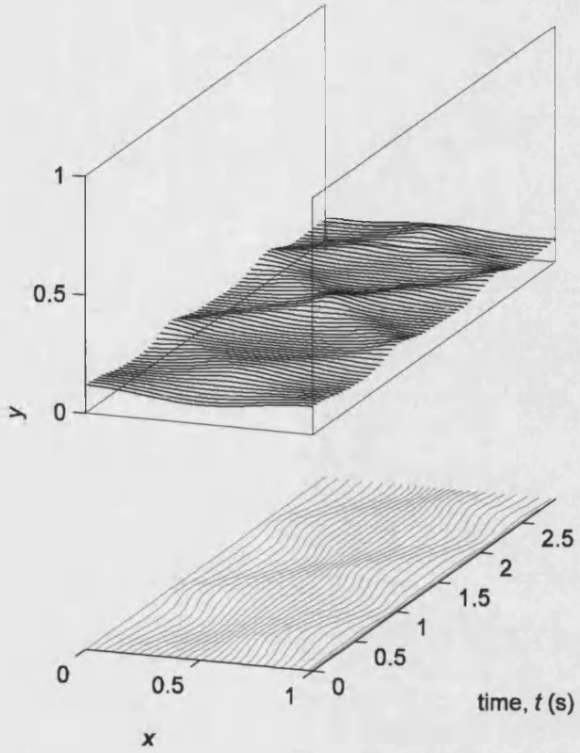


Figure 6.24. Succession of free surface profiles plotted at intervals of 0.05s, for the wave in Figure 6.21. The horizontal movement of the surface markers is shown at the bottom of the plot.

7. Conclusions & Recommendations

When attempting to model the behaviour of irrotational water waves, boundary-integral type methods are initially very appealing due to their ease of formulation and cheap computational costs. For these valid reasons, they have been extensively applied with considerable success. Problems arise when an extension of the code to deal with viscous flows is sought. In the work presented in this thesis, a method was devised to leave the possibility of this future development open. At the same time, it was sought to preserve the low computational costs afforded by boundary integral methods.

Accordingly, a fast grid generation algorithm with variable refinement has been developed in this research. A multigrid strategy has been optimised for hierarchical Cartesian grids and successfully employed to accelerate the convergence of the numerical solution. The fully Lagrangian surface markers have been successfully coupled with the fixed grid through a novel approach to calculate velocities at the free surface. These aspects are discussed in more detail in the following sections along with recommendations on uses and extensions to the proposed scheme.

7.1. Grid Generation

To restrict the burden on computational resources, irregular meshes were discarded due to their high generation costs and associated hunger for storage. Nevertheless, variable refinement is a desired property if accuracy is to be achieved without recourse to vast numbers of grid cells. Hierarchical Cartesian meshes proved well-suited in offering a balance between these two requirements. In Chapter 6, quadtrees were shown to provide a similar level of accuracy to a regular grid whilst using a tenth of the number of cells (cf. Figures 6.5 & 6.8). Their generation was also proved efficient, accounting for only 2% of total computational time per time step (cf. Figure 6.11) when a standard solver was used to simulate a standing wave.

7.2. Multigrid Strategy

Due to their tree-like structure, quadtrees are ideally suited to multigrid acceleration. It is in this field that the quadrilateral topology and hierarchical structure of quadtrees comes to the fore. Whilst the former allows simple and thus efficient grid transfer procedures, the latter permits very fast generation of coarser multigrid levels without incurring significant computational costs.

Two coarsening schemes were compared. The existing technique of Gáspár *et al.* (1991) (SCC) afforded savings in computational time only when a sufficient number of multigrid levels was used. When few multigrid levels were employed, however, computational times increased. By comparison, the global coarsening scheme proposed here always yielded faster convergence regardless of the number of multigrid levels used: its performance improved CPU times by 68-90% and outperformed the SCC method for any number of multigrid levels. This improvement is of special relevance for quadtrees with a considerable difference between large and small cell sizes (i.e. when $l_b << l_m$).

7.3. Lagrangian-Eulerian Coupling

The quadrilateral shape of Cartesian grid cells can cause problems when the fluid domain is bound by curved boundaries. This is clearly the case when water waves are concerned. This issue is prominent in two stages of the adopted finite difference solution method: a) in the solution of the velocity potential field where surface cells must use the potential values at the surface in their discretisation expressions; and b) in the computation of fluid velocities at surface markers.

In the first instance, the general method of unknown coefficients was employed to discretise the governing equations at these cells at each time step. Although some inaccuracies were expected due to the varying distances between markers and neighbouring nodes to surface cell centres, the method proved sufficiently accurate, once a limit was imposed on the minimum distance between the centres of Eulerian cells and surface markers.

The calculation of free surface velocities proved more susceptible to error in the initial stages of this research. Existing methods used with fitted meshes, such as least squares approximations, proved inadequate to the Lagrangian-Eulerian approach employed here. More than providing highly incorrect values, such methods yielded approximations of markedly different accuracy order for *consecutive* surface markers. This resulted in a localised degradation of the surface profile, which in a method such as this, tended to be amplified rather than smoothed out as the simulation time advances. To overcome this problem, the velocity calculation procedure of Section 4.4.3 was devised. Instead of Cartesian components, velocities are computed from normal and tangential parts. Whilst the latter is computed from adjacent surface marker values, the normal velocity is computed from points at a fixed distance inside the domain. This involves the additional calculation of the

velocity potential at these points which is achieved in the same way as for surface cells. The method has proved robust and accurate, with limited need for smoothing of the surface profile and showing good agreement with other methods in the Det Norske Veritas comparative study.

7.4. General Recommendations

In this final section, possible extensions to the existing code are considered in four aspects: co-ordinate transformations; the simulation of three-dimensional cases; the study of viscous flows; and the ability to deal with submerged and surface-piercing rigid bodies.

7.4.1. Co-ordinate Transformations

In recent months, the work of Frandsen & Borthwick (2003) has shown the advantages of employing a σ-transformation in the simulation of steep non-overturning waves. The method is easy to implement in the present code and has the significant advantage of eliminating the need for smoothing of the free surface. It is likely to render the scheme more robust and its implementation should not affect the use of the multigrid strategy.

7.4.2. Extension to Three Dimensions

In the development of the present method, the extension to three-dimensional cases and to viscous flows has been borne in mind. The generation of three-dimensional hierarchical Cartesian grids, called octrees, follows the same principles and techniques exposed in Chapter 3, and provides much the same advantages as quadtrees with respect to speed of grid generation. An example of an octree arrangement is displayed in Figure 7.1. Similarly, the storage, retrieval and manipulation of grid information can be performed with procedures akin to those described. For example, since

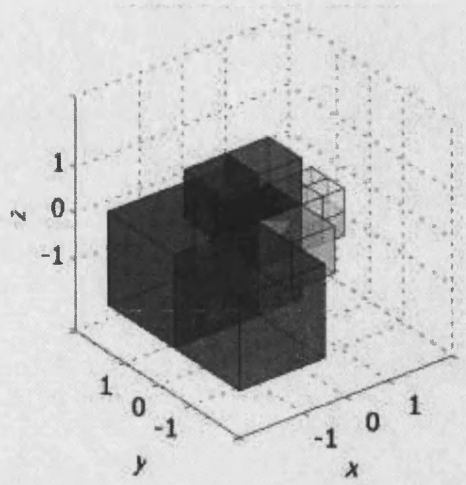


Figure 7.1 Example of octree arrangement

octrees are generated by recursive subdivision of hexahedra elements into eight hexahedra children, the numbering system is changed to a base 8 system in which equation (3.1) is replaced by,

$$n_c^i = 8n_p + i,$$

where $i \in \{1,2,3,\ldots,8\}$ is the position of the child cell within its parent.

The finite discretisation techniques described in Chapter 4 can be readily extended to three dimensions by including the partial derivatives of the velocity potential with respect to the third spatial variable. The treatment of surface cells need not therefore change. Similarly, the velocity calculation at the free surface can be performed in a similar fashion to the method proposed in Section 4.4.3, bearing in mind that two tangents to the free surface will exist, due to the added dimension, from which the surface normal should then be computed.

With respect to the generation of multigrid levels dealt with in Chapter 5, the global coarsening scheme proposed should yield even better results for three-dimensional cases as the scope for convergence acceleration is greater due to the larger number of cells employed.

The part of the algorithm which will require more care if an extension to three dimensions is sought is the ordering of the surface markers. Instead of a chain of Lagrangian particles, a free moving two-dimensional Lagrangian mesh would have to be tracked, with the correct ordering of markers maintained throughout the simulation. More refined techniques for the smoothing of the free surface are likely to be required, possibly coupled with modifications in the formulation to restrict excessive movement of the markers from their projection on the xy-plane.

7.4.3. Extension to Viscous Flows

The fact that the whole fluid domain is discretised in the method presented herein permits an extension of this work to the analysis of viscous flows, whilst preserving many essential components of the algorithm, with most of the work required being devoted to altering the solver routines to deal with the Navier-Stokes equations, as opposed to the Laplace equation. Multigrid iterations have been shown to provide good convergence acceleration for the simulation of viscous flows. For example, Darwish *et al.* (2003) have employed multigrid iterations to accelerate a solver based on Patankar (1980) SIMPLE method for incompressible flows. They use the multigrid strategy not only in the solution of the Poisson equation for the pressure, but also in the solution of the momentum equations and even the implementation of a k-ϵ turbulence method. Another example is the work of Wu *et al.* (1997) on a parallelized implementation of a multigrid algorithm for 2D incompressible flows.

It is thus expected that the savings afforded by the multigrid strategy in the method presented here can be translated into similar accelerations when viscosity is included.

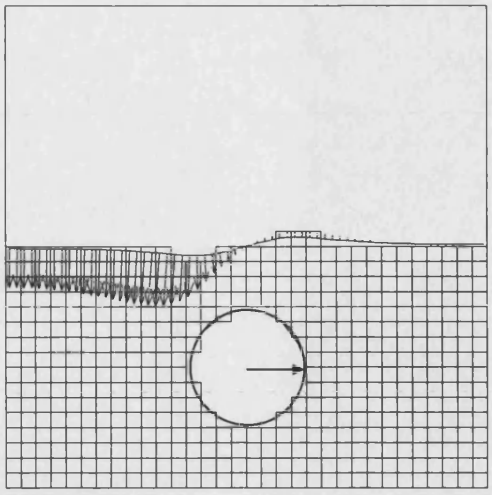


Figure 7.2. Disturbed free surface, and free surface velocities, as a result of submerged cylinder moving from left to right.

Additionally, the ability of hierarchical Cartesian grids to provide variable refinement would be beneficial to obtain a finer grid in areas of high vorticity, by coupling this quantity with the seeding point procedure of Section 3.1.

7.4.4. Interaction with Rigid Bodies

The algorithm can also be modified to accommodate the presence of solid objects in the domain. In what concerns grid generation, the points that define the rigid boundaries of these objects can be used as seeding points so as to yield higher refinement. The determination of which cells lie inside the solid object and are thus eligible to be excluded from the computation can be achieved by the geometric method of Section 3.5. The discretisation of grid cells in its vicinity can again be performed using the method of unknown coefficients of Section 4.2.4, employed for surface cells. Boundary conditions can be applied by creating boundary points on the body boundary at which the derivative of the potential in the direction normal to the body can be calculated. As an example, Figure 7.2 displays a preliminary study of the

effect of a moving submerged cylinder on an undisturbed surface (i.e. $\eta(x)=0$ for $0 \leq x \leq b$).

Appendix A. Discretisation Arrangements

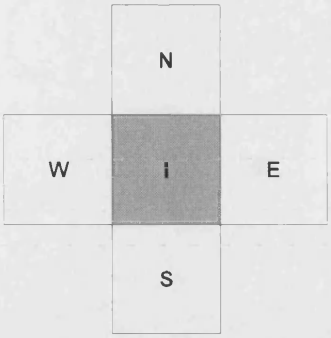


Figure A.1. Discretisation arrangement 1

$$\phi_I = \frac{\phi_W + \phi_E + \phi_S + \phi_N}{4} - \frac{(\Delta x)^2}{4} q$$

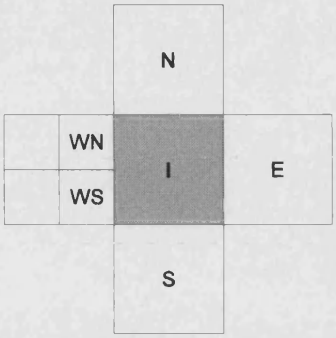


Figure A.2. Discretisation arrangement 2

$$\phi_I = \frac{6\phi_E + 5\phi_S + 5\phi_N + 4\phi_{WS} + 4\phi_{WN}}{24} - \frac{21(\Delta x)^2}{96} q$$

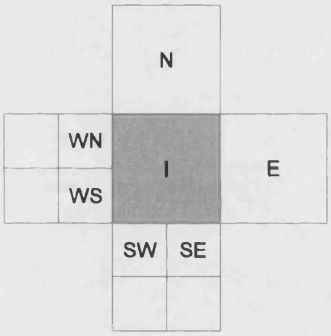


Figure A.3. Discretisation arrangement 3

$$\phi_I = \frac{3\phi_E + 3\phi_N + 2\phi_{WS} + 2\phi_{WN} + 2\phi_{SW} + 2\phi_{SE}}{14} - \frac{11(\Delta x)^2}{56} q$$

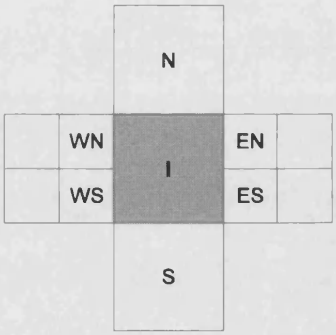


Figure A.4. Discretisation arrangement 4

$$\phi_I = \frac{\phi_{WSq} + \phi_{WN} + \phi_{EN} + \phi_{ES} + \phi_S + \phi_N}{6} - \frac{3(\Delta x)^2}{16} q$$

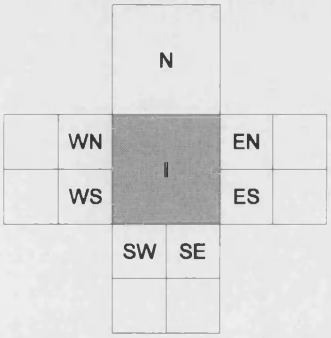


Figure A.5. Discretisation arrangement 5

$$\phi_I = \frac{6\phi_N + 5\phi_{WS} + 5\phi_{WN} + 5\phi_{ES} + 5\phi_{EN} + 4\phi_{SW} + 4\phi_{SE}}{34} - \frac{47(\Delta x)^2}{272} q$$

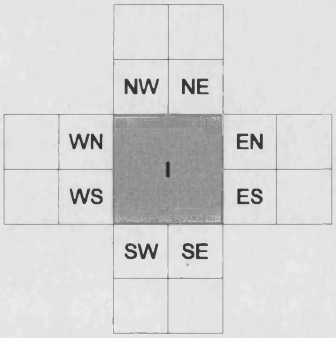


Figure A.6. Discretisation arrangement 6

$$\phi_I = \frac{\phi_{WS} + \phi_{WN} + \phi_{ES} + \phi_{EN} + \phi_{SW} + \phi_{SE} + \phi_{NW} + \phi_{NE}}{8} - \frac{4(\Delta x)^2}{5} q$$

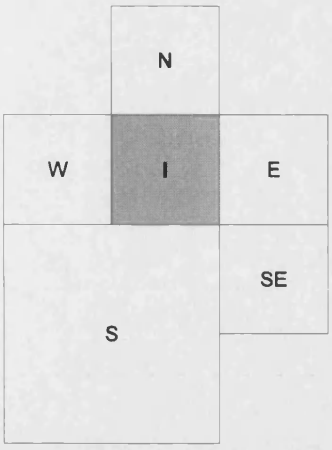


Figure A.7. Discretisation arrangement 7

$$\phi_I = \frac{9\phi_W + 9\phi_N + 8\phi_E + 4\phi_S + 3\phi_{SE}}{33} - \frac{7(\Delta x)^2}{22} q$$

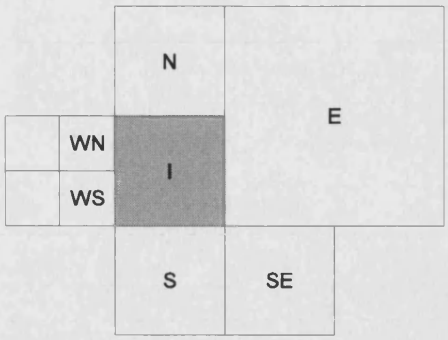


Figure A.8. Discretisation arrangement 8

$$\phi_I = \frac{124\phi_{WS} + 123\phi_N + 81\phi_S + 60\phi_{WN} + 56\phi_E + 54\phi_{SE}}{498} - \frac{189(\Delta x)^2}{664} q$$

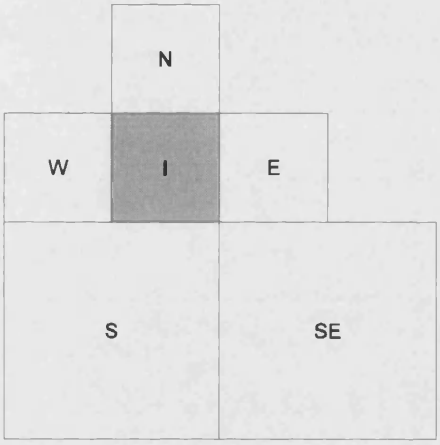


Figure A.9. Discretisation arrangement 9

$$\phi_I = \frac{6\phi_W + 6\phi_E + 6\phi_N + 3\phi_S + \phi_{SE}}{22} - \frac{15(\Delta x)^2}{44} q$$

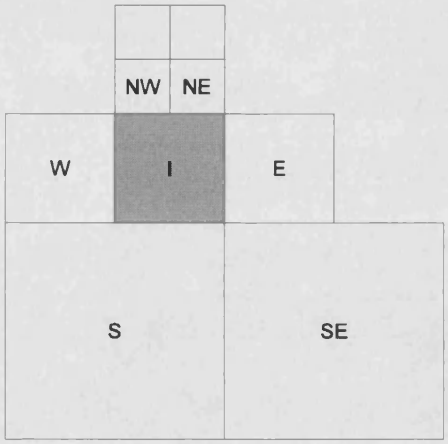


Figure A.10. Discretisation arrangement 10

$$\phi_I = \frac{5\phi_W + 5\phi_E + 4\phi_{NW} + 4\phi_{NE} + 3\phi_S + \phi_{SE}}{22} - \frac{27(\Delta x)^2}{88} q$$

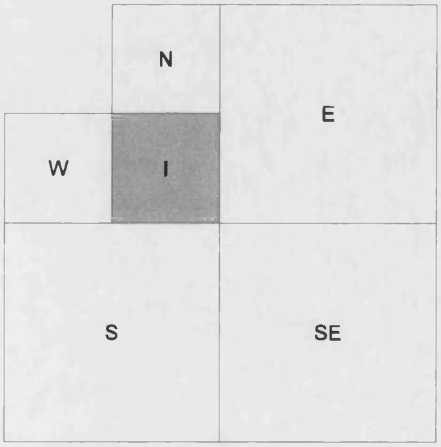


Figure A.12. Discretisation arrangement 12

$$\phi_I = \frac{6\phi_W + 6\phi_N + 3\phi_E + 3\phi_S + 2\phi_{SE}}{20} - \frac{9(\Delta x)^2}{20} q$$

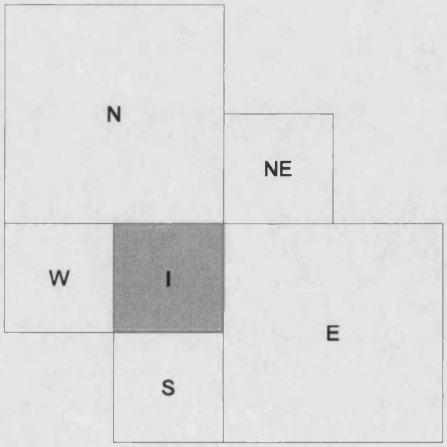


Figure A.11. Discretisation arrangement 11

$$\phi_I = \frac{5\phi_W + 5\phi_S + 3\phi_{NE} + 2\phi_E + 2\phi_N}{17} - \frac{13(\Delta x)^2}{34} q$$

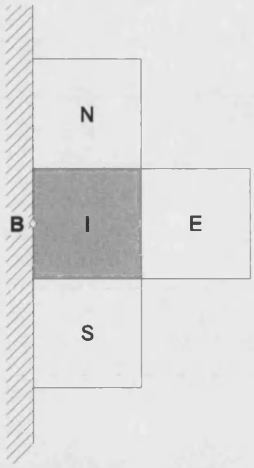


Figure A.13. Discretisation arrangement 13

$$\phi_I = \frac{8\phi_{BW} + 4\phi_E + 3\phi_S + 3\phi_N}{18} - \frac{(\Delta x)^2}{6} q$$

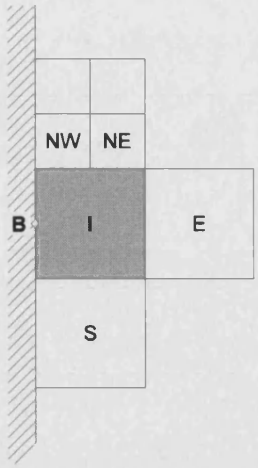


Figure A.14. Discretisation arrangement 14

$$\phi_I = \frac{20\phi_{BW} + 10\phi_E + 9\phi_S + 6\phi_{NW} + 6\phi_{NE}}{51} - \frac{21(\Delta x)^2}{136} q$$

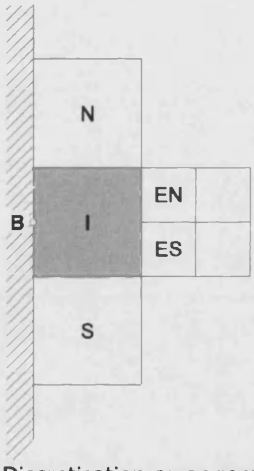


Figure A.15. Discretisation arrangement 15

$$\phi_I = \frac{24\phi_{BW} + 8\phi_{ES} + 8\phi_{EN} + 7\phi_S + 7\phi_N}{54} - \frac{5(\Delta x^2)}{36} q$$

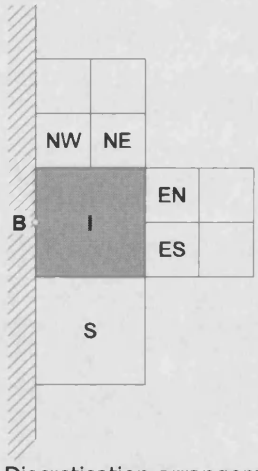


Figure A.16. Discretisation arrangement 16

$$\phi_I = \frac{44\phi_{BW} + 17\phi_{ES} + 14\phi_{NE} + 14\phi_S + 10\phi_{EN} + 7\phi_{NW}}{106} - \frac{55(\Delta x)^2}{424} q$$

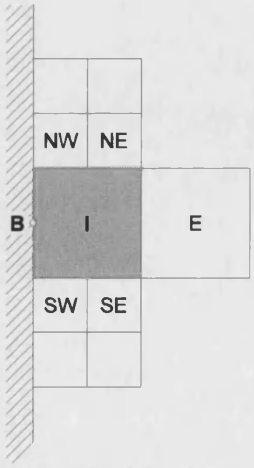


Figure A.17. Discretisation arrangement 17

$$\phi_I = \frac{8\phi_{BW} + 4\phi_E + 3\phi_{SW} + 3\phi_{SE} + 3\phi_{NW} + 3\phi_{NE}}{24} - \frac{9(\Delta x)^2}{64} q$$

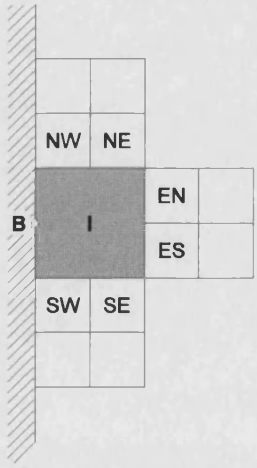


Figure A.18. Discretisation arrangement 18

$$\phi_I = \frac{8\phi_{BW} + 3\phi_{SW} + 3\phi_{NW} + 3\phi_{ES} + 3\phi_{EN} + 2\phi_{SE} + 2\phi_{NE}}{24} - \frac{(\Delta x)^2}{8} q$$

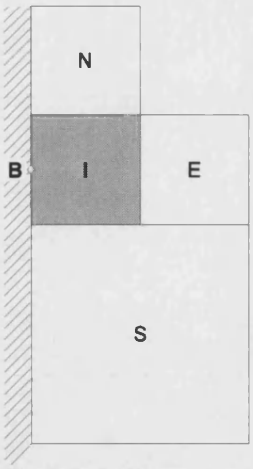


Figure A.19. Discretisation arrangement 19

$$\phi_I = \frac{32\phi_{BW} + 13\phi_E + 9\phi_N + 6\phi_S}{60} - \frac{3(\Delta x)^2}{16} q$$

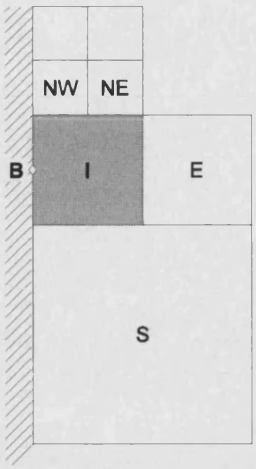


Figure A.20. Discretisation arrangement 20

$$\phi_I = \frac{12\phi_{BW} + 6\phi_{NE} + 3\phi_E + 2\phi_S - 2\phi_{NW}}{21} - \frac{9(\Delta x)^2}{56} q$$

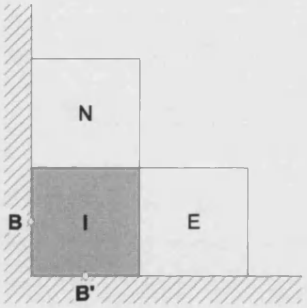


Figure A.21. Discretisation arrangement 21

$$\phi_I = \frac{2\phi_{BW} + 2\phi_{BS} + \phi_E + \phi_N}{6} - \frac{(\Delta x)^2}{8} q$$

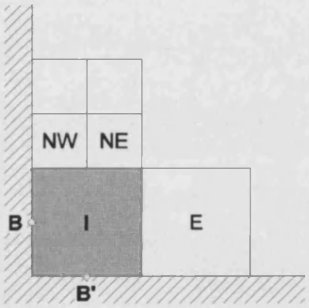


Figure A.22. Discretisation arrangement 22

$$\phi_I = \frac{18\phi_{BS} + 14\phi_{BW} + 7\phi_E + 6\phi_{NW} + 6\phi_{NE}}{51} - \frac{15(\Delta x)^2}{136} q$$

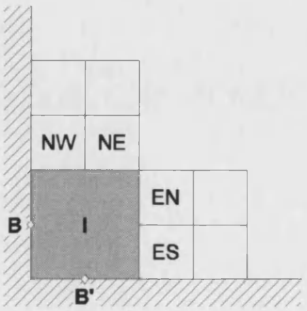


Figure A.23. Discretisation arrangement 23

$$\phi_I = \frac{3\phi_{BW} + 3\phi_{BS} + \phi_{ES} + \phi_{EN} + \phi_{NW} + \phi_{NE}}{10} - \frac{(\Delta x)^2}{10} q$$

Appendix B. Pseudo Code of Selected Procedures

List of variables and symbols

C, one-dimensional array containing tree path of cell
CHILD(I,Q), generation number of the child of cell I in quadrant position Q
I, cell generation number
L, division level index
LC, division level of cell
LEAF(I), logical flag, =TRUE if cell I is a leaf cell
N, cell reference number
XP,YP, x,y co-ordinates of point
XC,YC, x,y co-ordinates of cell
XD,YD. x,y co-ordiantes of root cell
WC, width/height of cell
WD, width/height of root cell

```
// Integer function that computes the division level of cell with reference
   number N

      integer function FLEVEL(N)

      FLEVEL = 0
      NAUX = N

// Successively apply equation (3.2) until reference number of root cell (0)
   is obtained
10    NAUX = int((NAUX - 1) / 4.0)
      if (NAUX.GT.0) then
         FLEVEL = FLEVEL + 1
         goto 10
      else
         if (N.NE.0) then
            FLEVEL = FLEVEL + 1
            return
         end if
      end if
      return

      end
```

Figure B.1. Code of function FLEVEL which determines the division level of cell with reference number N

```
// Subroutine that determines the tree path of a cell given its reference
   number, N. Result is returned in array C

      subroutine PATH(N,LC,C)

      LC = FLEVEL(N)
      do L = LC, 1, -1
         C(L) ← mod(N-1,4) + 1
         N = (N-C(L))/4
      end do
      return

      end
```

Figure B.2. Code of subroutine PATH which determines the tree path of cell with reference number N

```
// Function that calculates the size of cell with reference number N


      double precision function FWCELL(N,WD)

      FWCELL = WD
      LC = FLEVEL(N)

// Apply equation (3.4)
      FWCELL = WD / 2**LC
      return

      end
```

Figure B.3. Code of function FWCELL which determines the y co-ordinate of the centre of the cell with reference number N

```
// Function that calculates x co-ordinate of cell with reference number N


      double precision function FXCELL(N,XD)

      FXCELL = XD
      LC = FLEVEL(N)
      call PATH(N,LC,C)

// Apply equation (3.7a) to each quadrant position in the cell's tree path
      do L = 1, LC
         FXCELL = FXCELL + (2 * int((C(L) - 1),2) - 1)/2**(L+1)
      end do
      return

      end
```

Figure B4. Code of function FXCELL which determines the x co-ordinate of the centre of the cell with reference number N

```
// Function that calculates y co-ordinate of cell with reference number N


      double precision function FYCELL(N,XD)

      FYCELL = XD
      LC = FLEVEL(N)
      call PATH(N,LC,C)

// Apply equation (3.7b) to each quadrant position in the cell's tree path
      do L = 1, LC
         FYCELL = FYCELL + (2 * mod(C(L),2) - 1)/2**(L+1)
      end do
      return

      end
```

Figure B5. Code of function FYCELL which determines the y co-ordinate of the centre of the cell with reference number N

```
// Logical function that ascertains if point (XP,YP) is contained inside
   cell I with reference number N.

      logical function INSIDE(I,XP,YP)

      XC = FXCELL(N(I),XD)
      YC = FYCELL(N(I),YD)
      WC = FWCELL(N(I),SD)
      if (XC-SC/2)≤XP≤ (XC+SC/2) & (YC-WC/2)≤YP≤ (YC+WC/2) then
         INSIDE = .TRUE.
      else
         INSIDE = .FALSE.
      end if
      return

      end
```

Figure B.6. Code of function INSIDE which determines if point (XP,YP) is located inside cell with reference number N

```
// Recursive subroutine that determines which leaf cell contains a given
   point. Result is returned in variable IFOUND

      subroutine FIND_CELL_CONTAINING_P(I,XP,YP)

// If cell I has children, recursively recall this subroutine using the
   number of the child cell containing P as input
      if not(LEAF(I)) then
         if INSIDE(CHILD(I,1),XP,YP) then
            call FIND_CELL_CONTAINING_P(CHILD(I,1),XP,YP)
         elseif INSIDE(CHILD(I,2),XP,YP) then
            call FIND_CELL_CONTAINING_P(CHILD(I,2),XP,YP)
         elseif INSIDE(CHILD(I,3),XP,YP) then
            call FIND_CELL_CONTAINING_P(CHILD(I,3),XP,YP)
         elseif INSIDE(CHILD(I,4),XP,YP) then
            call FIND_CELL_CONTAINING_P(CHILD(I,4),XP,YP)
         end if

// If cell I is a leaf cell, the procedure is complete, and I is returned
      else
         IFOUND = I
         return
      end if

      end
```

Figure B.7. Code of recursive subroutine FIND_CELL_CONTAINING_P used to find the leaf cell containing a given point (XP,YP)

References

Addessio, F. L.; Carroll, D. E.; Dukowicz, J. K.; Harlow, F. H.; Johnson, J. N.; Kashiwa, B. A.; Maltrud, M. E. and Ruppel, H. M. (1986) CAVEAT: A Computer Code for Fluid Dynamics Problems with Large Distortion and Internal Slip, Los Alamos National Laboratory Report, LA-10613-MS

Amsden, A. A. and Harlow, F. H. (1970) The SMAC Method: A Numerical Technique for Calculating Incompressible Fluid Flows, Los Alamos National Laboratory Report, LA-4370

Andrillon, Y.; Doring, M.; Alessandrini, B. and Fant, P. (2002) Comparison between SPH and VOF Free Surface Flow Simulation, 5th Numer. Towing Tank Symposium, Pornichet, France

Armenio, V. (1997) An Improved MAC Method (SIMAC) For Unsteady High-Reynolds Free Surface Flows, *Int. J. Numer. Methods Fluids*, **24**, 185-214

Belytschko, T.; Krongauz, Y.; Dolbow, J. and Gerlach, C. (1998) On the Completeness of Meshfree Particle Methods, *Int. J. Numer. Mech. Engng.*, **43**, 785-819

Berger, M. J.; Aftosmis, M. and Adomavicious, G. (2001) Parallel Multigrid on Cartesian Meshes with Complex Geometry, Proc. Parallel CFD Conf. 2000, Trondheim, Norway, pp. 283-290

Birdsall, C. K. and Fuss, D. (1969) Clouds-in-Clouds, Clouds-in-Cells Physics for Many-Body Plasma Simulation, *J. Comp. Physics*, **3**, 494-511

Brackbill, J. U.; Kothe, D. B. and Zemach, C. (1992) A Continuum Method for Modeling Surface Tension, *J. Comp. Physics*, **100**, 335-354

Brandt, A. (1977) Multi-level Adaptive Solutions to Boundary Value Problems, *Math. of Comp.*, **31**, 333-390

Briggs, W. L. (1987) A Multigrid Tutorial, Society for Industrial and Applied Mathematics, Philadelphia

Butler, T. D. (1971) LINC Method Extensions, Proc. 5th Int. Conf. Fluid Dynamics, Twente, The Netherlands, pp. 435-440

Chen, L.; Garimella, S. V.; Reizes, J. A. and Leonardi, E. (1997) Motion of Interacting Gas Bubbles in a Viscous Liquid Including Wall Effects and Evaporation, *Numer. Heat Transfer, Part A*, **31**, 629-654

Chen, S.; Johnson, D. B. and Raad, P. E.(1991) The Surface Marker Method, Proc. 1st Int. Conf. Computational Modelling Free and Moving Boundary Problems, Vol. I - Fluid Flow, Southampton, UK, pp. 223-234

Chen, S.; Johnson, D. B. and Raad, P. E.(1995) Velocity Boundary Conditions for the Simulation of Free Surface Fluid Flow, *J. Comp. Physics*, **116**, 262-276

Chern, I. L.; Glimm, J.; McBryan, O.; Plohr, B. and Yaniv, S.(1986) Front Trackinf for Gas Dynamics, *J. Comp. Physics*, **62**, 83-110

Chorin, A. J.(1980) Flame Advection and Propagation, *J. Comp. Physics*, **35**, 1-11

Cointe, R.(1989) Quelques Aspects de la Simulation Numérique d'un Canal à Houle, PhD Thesis, École Nationale des Ponts et Chaussées, Paris

Cointe, R.; Geyer, P.; King, B.; Molin, B. and Tramoni, M.(1991) Nonlinear and Linear Motions of a Rectangular Barge in a Perfect Fluid, Proc. 18th Symposium Naval Hydrodynamics, Univ. Michigan, Ann Arbor, USA, pp. 85-98

Colagrossi, A. and Landrini, M.(2002) Numerical Simulation of Two-Phase Flows by Smoothed Particle Hydrodynamics, 5th Numer. Towing Tank Symposium, Pornichet, France

Colicchio, G.; Landrini, M. and Chaplin, J.(2002) Level-Set Simulation of the Vortical Flow Generated by a Surface-Piercing Body, 5th Numer. Towing Tank Symposium, Pornichet, France

Crowley, W. P. (1971) FLAG: A free-Lagrange Method for Numerically Simulating Hydrodynamic Flows in Two-Dimensions, Proc. 5th Int. Conf. Fluid Dynamics, Twente, The Netherlands, pp. 37-43

Daly, B. J.; Harlow, F. H. and Welch, J. E.(1964) Numerical Fluid Dynamics Using the Particle-and-Force Method, Los Alamos Laboratory Report, LA-3144

Darwish, M.; Moukalled, F. and Sekar, B.(2003) A Robust Multi-Grid Pressure-Based Algorithm for Multi-Fluid Flow at All Speeds, *Int. J. Numer. Methods Fluids*, **41**, 1221-1251

de Sosa, J.(1676) Noticia de la Gran Casa de los Marqueses de Villafranca y su Parentesco con las Mayores de Europa en el Árbol Genealógico de la Ascendencia del Excelentissimo Señor D. Fadrique de Toledo Osorio, Novello de Bonis, Naples

De Zeeuw, D. and Powell, K. G.(1993) An Adaptively Refined Cartesian Mesh Solver for the Euler Equations, *J. Comp. Physics*, **104**, 56-68

Delaunay, B. (1934) Sur la Sphere Vide, *Bull. Acad. Sci. USSR (VII), Classe Sci. Mat. Nat.*, 793-800

Dold, J. W.(1992) An Efficient Surface-Integral Algorithm Applied to Unsteady Gravity Waves, *J. Comp. Physics*, **103**, 90-115

Dommermuth, D. G. and Yue, D. K. P.(1987) Numerical Simulations of Nonlinear Axisymmetric Flows with a Free Surface, *J. Fluid Mech.*, **178**, 195-219

Dommermuth, D. G.; Yue, D. K. P.; Lin, W. M.; Rapp, R. J.; Chan, E. S. and Melville, W. K. (1988) Deep-Water Plunging Breakers: A Comparison Between Potential Theory and Experiments, *J. Fluid Mech.*, **189**, 423-442

Dukowicz, J. K. (1981) Lagrangian Fluid Dynamics Using the Voronoi-Delaunay Mesh, in Numerical Methods for Coupled Problems, Pineridge, Swansea, pp. 82-104

Dukowicz, J. K. (1984) Conservative Rezoning (Remapping) for General Quadrilateral Meshes, *J. Comp. Physics*, **54**, 411-424

Dukowicz, J. K. and Kodis, J. M.(1985) Accurate Conservative Remapping (Rezoning) for Arbitrary Lagrangian-Eulerian Computations, Los Alamos National Laboratory Report, LA-UR-2646

Dussan, V. E. B.(1979) On The Spreading Of Liquid On Solid Surfaces, *Annu. Rev. Fluid Mech.*, **11**, 371-400

Eatock Taylor, R. and Wu, GX. (1986) A Comparison of Localized Finite Element Formulations for Two-Dimensional Wave Diffraction and Radiation Problems, *Int. Shipbldg Prog.*, **33**, 2-9

Evans, A. (1993) Mesh Adaptivity in Compressible Flow, PhD Thesis, University College of Swansea

Faltinsen, O. M.(1977) Numerical Solution of Transient Free-Surface Motion Outside or Inside Moving Bodies, Proc. 2nd Int. Conf. Numer. Ship Hydrodynamics, University California Berkeley, pp. 347-357

Fedorenko, R. P.(1964) The Speed of Convergence of an Iterative Process, *Comp. Math. Phys.*, **4**, 227-235

Ferrant, P. (1997) Simulation of Strongly Nonlinear Wave Generation and Wave-Body Interactions Using a 3-D MEL Model, Proc. 21st Symposium Naval Hydrodynamics, pp. 93-109

Fletcher, C. A. J. (1991) Computational Techniques for Fluid Dynamics, 2nd ed., Springer-Verlag, Berlin

Floryan, J. M. and Rasmussen, H. (1989) Numerical Methods For Viscous Flows With Moving Boundaries, *Annu. Mech. Rev.*, **42**, 12, 323-341

Frandsen, J. B. and Borthwick, A. G. L. (2003) Simulation of Sloshing Motions in Fixed and Vertically Excited Containers Using a 2-D Inviscid σ-transformed Finite Difference Solver, *J. Fluids Struct.*, **18**, 197-214

Fritts, M. J.; Crowley, W. P. and Trease, H. E. (1985) The Free Lagrange Method, *Notes in Physics*, **238**, Springer-Verlag, Berlin

Gáspár, C. and Józsa, J. (1991) A Coupled Lagrangian Particle Tracking and Quadtree-Based Adaptive Multigrid Method with Application to Shear LAyer Evolution, Proc. XXIV Int. Assoc. of Hydraulic Engineering and Research Congress, Madrid, Spain

Gáspár, C.; Józsa, J. and Simbierowicz, P. (1991) Lagrangian Modelling of the Convective Diffusion Problem Using Unstructured Grids and Multigrid Technique, Proc. 1st Int. Conf. Water Pollution (Modelling, Measuring and Prediction), Southampton, UK

Gáspár, C. and Simbierowicz, P. (1992) Difference Schemes in Tree-Structured Multigrid Context, Proc. 9th Int. Conf. Comp. Methods Water Resources, Denver, USA

Gingold, R. A. and Monaghan, J. J. (1977) Smoothed Particle Hydrodynamics: Theory and Application to Non-Spherical Stars, *Mon. Not. R. Astron. Soc.*, **181**, 375-389

Greaves, D. M. (1995) Numerical Modelling of Laminar Separated Flows and Inviscid Steep Waves Using Adaptive Hierarchical Meshes, DPhil Thesis, University of Oxford, UK

Greaves, D. M.; Borthwick, A. G. L.; Wu, G. and Eatock Taylor, R. (1997) A Moving Boundary Finite Element for Fully Nonlinear Wave Simulations, *J. Ship Research*, **41**, 3, 181-194

Greenhow, M.; Vinje, T.; Brevig, P. and Taylor, J. (1982) A Theoretical and Experimental Study of the Capsize of Salter's Duck in Extreme Waves, *J. Fluid Mech.*, **118**, 221-239

Hackbusch, W. (1985) Multi-grid Methods and Applications, Springer-Verlag, Berlin

Hackbush, W. (1985) Multi-grid Methods and Applications, Springer-Verlag, Berlin

Harlow, F. H. (1964) The Particle-in-Cell Computing Method for Fluid Dynamics, in Methods in Computational Physics, Academic Press, New York, pp. 319-343

Harlow, F. H. and Welch, J. E. (1965) Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface, *Physics Fluids*, **8**, 12, 2182-2189

Hino, T. and Hirata, N. (2002) Unstructured Multigrid Method for Ship Flows, 5th Numer. Towing Tank Symposium, Pornichet, France

Hirt, C. W.; Amsden, A. A. and Cook, J. L. (1974) An Arbitrary Lagrangian-Eulerian Computing Method for all Speeds, *J. Comp. Physics*, **14**, 227-253

Hirt, C. W.; Cook, J. L. and Butler, T. D. (1970) A Lagrangian Method for Calculating the Dynamics of an Incompressible Fluid with Free Surface, *J. Comp. Physics*, **5**, 103-124

Hirt, C. W. and Nichols, B. D. (1981) Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries, *J. Comp. Physics*, **39**, 201-225

Hockney, R. W. and Eastwood, J. W. (1981) Computer Simulation Using Particles, McGraw-Hill, New York

Iafrati, A. and Campana, E. F. (2003) A Domain Decomposition Approach to Compute Wave Breaking (Wave-Breaking Flows), *Int. J. Numer. Methods Fluids*, **41**, 419-445

Isaacson, M. d. St. Q. (1982) Nonlinear-Wave Effects on Fixed and Floating Bodies, *J. Fluid Mech.*, **120**, 267-281

Jeong, J. H. and Yang, G. (1998) Finite Element Analysis of Transient Fluid Flow with Free Surface Using VOF (Volume-Of-Fluid) Method and Adaptive Grid, *Int. J. Numer. Methods Fluids*, **26**, 1127-1154

Jeong, J. H. and Yang, G. (1999) Three-Dimensional Finite Element Analysis of Transient Fluid Flow with Free-Surface Using Marker Surface Method and Adaptive Grid Refinement, *Int. J. Numer. Methods Fluids*, **29**, 657-684

Józsa, J. and Gáspár, C. (1992) Fast, Adaptive Approximation of Window-Induced Horizontal Flow Patterns in Shallow Lakes Using Quadtree-Based Multigrid Method, Proc. 9th Int. Conf. Comp. Methods Water Resources, Denver

Lafaurie, B.; Nardone, C.; Scarvodelli, R.; Zaleski, S. and Zanetti, G. (1994) Modelling Merging and Fragmentation in Multiphase Flows with SURFER, *J. Comp. Physics*, **113**, 134-147

Landrini, M.; Colagrossi, A. and Tulin, M. (2001) Breaking Bow and Stern Waves: Numerical Simulations, Proc. 16th Int. Workshop Water Waves Floating Bodies, Hiroshima

Leonard, B. P.(1979) A Stable and Accurate Convective Modelling Procedure Based on Quadratic Upstream Interpolation, *Comp. Methods in Appl. Mech and Eng.*, **19**, 59-98

Lin, W. M.; Newman, J. N. and Yue, D. K. P.(1984) Nonlinear Forced Motion of Floating Bodies, Proc. 15th Symposium Naval Hydrodynamics, Hamburg, Germany, pp. 33-49

Longuet-Higgins, M. S.(1953) On the Decrease of Velocity with Depth in an Irrotational Water Wave, *Proc. Camb. Phil. Soc.*, **49**, 552-560

Longuet-Higgins, M. S. and Cokelet, E. D.(1976) The Deformation of Steep Surface Waves on Water; I. A Numerical Method of Computation, *Proc. R. Soc. London A*, **350**, 1-26

Lucy, L. B. (1977) A Numerical Approach to the Testing of the Fission Hypothesis, *Astron. J.*, **82**, 1013-1024

Mercer, G. N. and Roberts, A. J.(1992) Standing Waves in Deep Water: Their Stability and Extreme Form, *Physics Fluids*, **A4(2)**, 259-269

Miyata, H. (1986) Finite-Difference Simulation of Breaking Waves, *J. Comp. Physics*, **65**, 179-214

Miyata, H. and Yamada, Y.(1992) A Finite Difference Method for 3-D Flows about Bodies of Complex Geometry in Rectangular Co-ordinate Systems, *Int. J. Numer. Methods Fluids*, **14**, 1261-1287

Monaghan, J. J.(1992) Smoothed Particle Hydrodynamics, *Annu. Rev. Astron. Astrophys.*, **30**, 543-574

Monaghan, J. J.(1994) Simulating Free Surface Flows with SPH, *J. Comp. Physics*, **110**, 399-406

Nestegård, A.(1994) Comparative Study of Fully Non-Linear Wave Simulation Programs, Det Norske Veritas Report, 94-2041

Nichols, B. D.; Hirt, C. W. and Hotchkiss, R. S.(1981) A Fractional Volume of Fluid Method for Free Boundary Dynaimcs, Proc. 7th Int. Conf. Numer. Methods Fluid Dynamics, Stanford, USA, pp. 304-309

Noh, W. F. and Woodward, P.(1976) SLIC (Simple Line Interface Calculation), in Proc. 5th Int. Conf. Fluid Dynam., *Lecture Notes in Physics,* **59**, Springer-Verlag, Berlin, pp. 330-340

Osher, S. and Sethian, J. A.(1988) Fronts Propagating with Curvature-Dependent Speed Algorithms Based on Hamilton-Jacobi Formulations, *J. Comp. Physics*, **79**, 12-49

Park, J. C.; Kim, M. H. and Miyata, H.(1999) Fully Non-Linear Free Surface Simulations by a 3D Viscous Numerical Wave Tank, *Int. J. Numer. Methods Fluids*, **29**, 685-703

Patankar, S. V.(1980) Numerical Heat Transfer and Fluid Flow, McGraw-Hill, New York

Peregrine, D. H.(1972) Flow Due to Vertical Plate Moving in a Channel, Unpublished work, Department of Mathematics, University of Bristol

Pert, G. J.(1983) Quasi-Lagrangian Rezoning of Fluid Codes Maintaining an Orthogonal Mesh, *J. Comp. Physics*, **49**, 1-43

Price, W. G. and Tan, M.(1992) Fundamental Viscous Solutions or 'Transient Oseenlets' Associated with a Body Manoeuvering in a Viscous Fluid, *Proc. R. Soc. London A*, **438**, 447-466

Rienecker, M. M. and Fenton, J. D.(1981) A Fourier Approximation Method for Steady Water Waves, *J. Fluid Mech.*, **104**, 119-137

Russo, G. and Smereka, P.(2000) A Remark on Computing Distance Functions, *J. Comp. Physics*, **163**, 51-67

Samet, H. (1982) Neighbor Finding Techniques for Images Represented by Quadtrees, *Computer Graphics and Image Processing*, **18**, 37-57

Samet, H. (1990) Applications of Spatial Data Structures, Addison-Wesley, Reading, USA

Saville, G. (1977) Computer Simulation of the Liquid-Solid-Vapour Contact Angle, *J. Chem. Soc. Faraday Trans.*, **73**, 1122-1132

Schmidt, R. J. (1991) Adaptive Quadtree Discretization for Fluuid Flow Problems, PhD Thesis, Rensselaer Polytechnic Institute, New York

Sussman, M.; Smereka, P. and Osher S. (1994) A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow, *J. Comp. Physics*, **114**, 146-159

Tanizawa, K.(1995) A Nonlinear Simulation Method of 3-D Body Motions in Waves, Proc. 10th Int. Workshop Water Waves Floating Bodies, Oxford, UK, pp. 235-239

Tanizawa, K.(1996) A Nonlinear Simulation Method of 3-D Body Motions in Waves Extended Formulation for Multiple Fluid Domains, Proc. 11th Int. Workshop Water Waves Floating Bodies, Hamburg, Germany

Turnbull, M. S.(1999) The Numerical Modelling of Steep Waves Interacting with Structures, DPhil Thesis, University of Oxford, UK

Ubbink, O. (1997) Numerical Prediction of Two Fluid Systems with Sharp Interfaces, DPhil Thesis, Imperial College of Science, Technology and Medicine, UK

Udaykumar, H. S.; Mittal, R.; Rampunggoon, P. and Khanna, A.(2001) A Sharp Interface Cartesian Grid Method .for Simulating Flows with Complex Moving Boundaries, *J. Comp. Physics*, **174**, 345-380

Unverdi, S. O. and Tryggvason, G.(1992) A Front-Tracking Method for Viscous, Incompressible, Multi-Fluid Flows, *J. Comp. Physics*, **100**, 25-37

van Dommelen, L. and Rundensteiner, E. A.(1989) Fast, Adaptive Summation of Point Forces in the Two-Dimensional Poisson Equation, *J. Comp. Physics*, **83**, 126-147

Varga, R. S.(1962) Matrix Iterative Analysis, Prentice-Hall, London

Vinje, T. and Brevig, P.(1981) Numerical Simulation of Breaking Waves, *Adv. Water Resources*, **4**, 77-82

von Strodonitz, S. K.(1904) Ahnentafel-Atlas. Ahnentafeln zu 32 Ahnen der Regenten Europas und ihrer Gemahlinnen, Berlin

Voronoi, G. (1908) Nouvelles Applications des Paramètres Continus a la Theorie des Formes Quadratiques, Deuxième Memoire Recherches sur les Parallelloedres Primitiff, *Z. Reine Angew. Math.*, **134**, 198-203

Warren, M. S. and Salmon, J. K.(1994) A Portable Parallel Particle Program, *Comp. Phys. Comm.*, **87**, 266-290

Wesseling, P.(1988) Cell-Centred Multigrid for Interface Problems, *J. Comp. Physics*, **79**, 85-91

Wesseling, P. (1992) An Introduction to Multigrid Methods, John Wiley & Sons, Chichester

Wu, G.-X. and Eatock Taylor, R.(1987) Hydrodynamic Forces on Submerged Oscillating Cylinders at Forward Speed, *Proc. R. Soc. London A*, **414**, 149, 170

Wu, G.X. and Eatock Taylor, R. (1994) Finite Element Analysis of Two-Dimensional Non-Linear Transient Water Waves, *Applied Ocean Res.*, **16**, 363-372

Wu, G.X. and Eatock Taylor, R. (1996) Transient Motion of a Floating Body in Steep Water Waves, Proc. 11th Int. Workshop Water Waves Floating Bodies, Hamburg

Wu, G.X.; Ma, Q.W. and Eatock Taylor, R. (1997) Analysis of Interactions Between Nonlinear Waves and Bodies by Domain Decomposition, Proc. 21st Symposium Naval Hydrodynamics, pp. 110-119

Wu, J.; Ritzdorf, H.; Oosterlee, K.; Steckel, B. and Schuller, A. (1997) Adaptive Parallel Multigrid Solution of 2D Incompressible Navier-Stokes Equations, *Int. J. Numer. Methods Fluids*, **24**, 9, 875-892

Yeung, R. W. and Ananthakrishnan, P. (1989) Solution of Nonlinear Water-Wave and Water-Body Interaction Problems Using a New Boundary-Fitted Coordinates Method, Proc. 4th Int. Workshop Water Waves Floating Bodies, Øystese, Norway

Yeung, R. W. and Vaidhanathan, M. (1990) Nonlinear Wave Diffraction Over Submerged Obstacles, Proc. 5th Int. Workshop Water Waves Floating Bodies, Manchester, UK

Yiu, K. F. C.; Greaves, D. M.; Cruz, S.; Saalehi, A. and Borthwick, A. G. L. (1996) Quadtree Grid Generation: Information Handling, Boundary Fitting and CFD Applications, *Computers & Fluids*, **25**, 8, 759-769

Young, D. P.; Melvin, R. G.; Bieterman, M. B.; Johnson, F. T.; Samant, S. S. and Bussolett, J. E. (1991) A Locally Refined Rectangular Grid Finite Element Method: Application to Computational Fluid Dynamics and Computational Physics, *J. Comp. Physics*, **92**, 1-66

Youngs, D. L. (1982) Time-Dependent Multi-Material Flow with Large Fluid Distortion, in Numerical Methods for Fluid Dynamics, Academic Press, New York, pp. 273-285

Zhou, Z. and Gu, M. (1991) A Numerical Research of Nonlinear Body-Wave Interactions, Proc. 18th Symposium Naval Hydrodynamics, pp. 103-117